MEME: Generating RNN Model Explanations via Model Extraction

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recurrent Neural Networks (RNNs) have achieved remarkable performance on a range of tasks. A key step to further empowering RNN-based approaches is improving their explainability and interpretability. In this work we present MEME: a model extraction approach capable of approximating RNNs with interpretable models represented by human-understandable concepts and their interactions. We demonstrate how MEME can be applied to two multivariate, continuous data case studies: Room Occupation Prediction, and In-Hospital Mortality Prediction. Using these case-studies, we show how our extracted models can be used to interpret RNNs both locally and globally, by approximating RNN decision-making via interpretable concept interactions.

1 Introduction

Deep Learning (DL) has achieved groundbreaking results in a wide range of fields [9], and is currently an active research area of Artificial Intelligence (AI). Recurrent Neural Networks (RNNs) are DL models tailored for processing time-series data, and have been successfully applied in domains relying on temporal data processing, including healthcare [8], Natural Language Processing (NLP) [7], video processing [10], and many others [30].

A central challenge faced by most DL approaches is their lack of interpretability: the black-box nature of most DL models makes it very difficult to directly understand their behaviour. This often leads to a lack of trust in such models. As a result, it is challenging to apply and regulate these models in safety-critical applications, such as healthcare. This lack of interpretability also makes it challenging to extract knowledge from such models, which prohibits us from better understanding their corresponding tasks/domains. As a result, there has recently been a dramatic increase in research on Explainable AI (XAI), focusing on improving explainability/interpretability of DL systems [11].

In particular, one type of XAI approach is *model extraction* (also referred to as *model translation*) [12]. Model extraction techniques use rules [37], decision trees [38], or other more readily explainable models [39] to approximate complex models, in order to study their behaviour. Intuitively, statistical properties of a complex model should be reflected in an extracted model, provided approximation quality (referred to as *fidelity*) is high enough.

In this work, we present MEME: an approach to RNN (M)odel (E)xplanation via (M)odel (E)xtraction.[1] Using MEME, we approximate RNN models with models represented via human-understandable concepts and their interactions. A diagrammatic summary of our approach is given in

[1] *Meme* refers to a system's behaviour/knowledge passed from one individual to another by imitation, akin to how an extracted model extracts knowledge from an original one via approximation. https://en.wikipedia.org/wiki/Meme

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

Figure 1 and Figure 2, and will be discussed in more detail in Section 3. All relevant code is publicly available, and can be found in our open-source repository.[2]

To summarize, we make the following contributions in this work:

- Present a novel model extraction approach (MEME), capable of approximating RNN models with interpretable models represented by human-understandable concepts and their interactions. To the best of our knowledge, this is the first time concept extraction approaches have been used with RNNs and time-series tasks.
- Quantitatively evaluate MEME using two *multivariate, continuous* data tasks (including a healthcare scenario), showing that MEME generates high-fidelity extracted models. To the best of our knowledge, this is the first time RNN model extraction approaches have been applied to multivariate, continuous data tasks.
- Qualitatively evaluate MEME using the two case studies, showing how extracted models produced by MEME can be inspected in order to achieve both *global* explainability (i.e., explaining overall model behaviour), and *local* explainability (explaining individual predictions) of RNN models.

2 Related Work

Concept Extraction Most existing work on post-hoc DL explanation methods (those focusing on explaining trained DL models) provides explanations by estimating the importance of input features, and typically focuses on local explanations [13]. However, these methods have been shown to have methodological and theoretical limitations [15, 14]. More recent approaches focus on *concept-based* explanations, presenting explanations in terms of human-understandable *concepts*, which are typically derived from the hidden layers of a DL model [16]. Crucially, existing concept extraction approaches focus predominantly on image recognition tasks, as opposed to time-series tasks, which is the focus of this work. Further details regarding concept-based explanations can be found in Appendix A.

RNN Model Extraction Several recent works have presented approaches to RNN model extraction, focusing predominantly on language-based modelling NLP tasks. For instance, work in [21, 20, 22] presents approaches for approximating RNNs via Finite State Automata (FSA), Probabilistic Deterministic Finite Automata (PDFA), or Discrete-Time Markov Chains (DTMC). Similarly to our work, these works generate extracted models by quantising the RNN hidden space, and approximating the RNN transition dynamics in this space. However, these works focus on tasks consisting of categorical, univariate data, whereas we consider tasks with multivariate continuous data. Furthermore, we associate the quantised states with human-understandable concepts, as will be discussed in Section 3.

RNN Explainability Another direction for RNN explainability focuses on using specialised model architectures [23, 24], or specialised model training procedures [25]. Predominantly, these approaches rely on *attention mechanisms* [23, 24], which require modification of the model architecture and/or model training procedure. In contrast, our approach is agnostic to the RNN architecture, and requires no modifications/retraining. Furthermore, it has recently been argued that "attention is not explanation", emphasising that in practice, it is often unclear what relationship exists between attention weights and model outputs [26].

3 Methodology

3.1 Setup

Without loss of generality, we consider pre-trained RNN models containing a recurrent layer l with m neurones, processing sequences of inputs $\mathbf{x}_t, \mathbf{x}_{t+1}, ...$, with multivariate, continuous input samples $\mathbf{x}_t \in \mathbb{R}^n$. By definition, recurrent layers retain a hidden state. Thus, an input inference and its layer l activations at a given timestep can be written as a function $f_{hidden} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^m$, such that the function f_{hidden} takes the next input data-point $\mathbf{x}_t \in \mathbb{R}^n$, the previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^m$, and outputs the next hidden state $\mathbf{h}_t \in \mathbb{R}^m$. In this work, we focus on binary classification tasks, in which

[2] Code will be released after paper review.

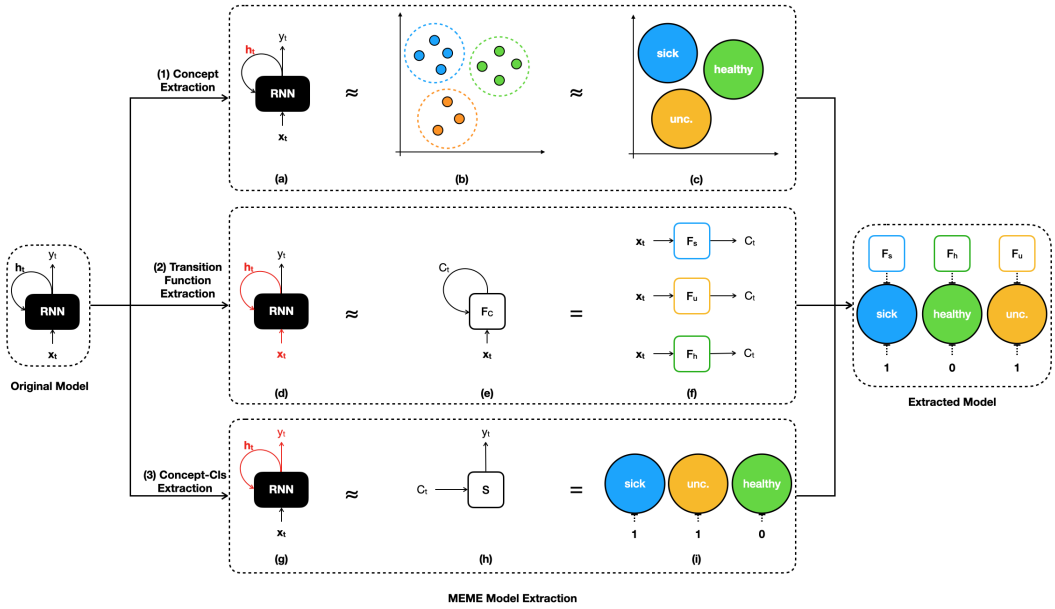


Figure 1: Given an RNN model, we: (1) approximate it's hidden space by a set of concepts. (2) approximate it's hidden space dynamics by a set of transition functions, one per concept. (3) approximate it's output behaviour by a concept-class mapping, specifying an output class label for every concept. For every step in (1)-(3), the parts of the RNN being approximated are highlighted in red. In (a)-(c) we cluster the RNN's training data points in their hidden representation (assumed to be two-dimensional, in this example), and use the clustering to produce a set of concepts (in this case: *sick*, *healthy* and *uncertain*, written as *unc.*). In (d)-(f) we approximate the hidden function of the RNN by a function F_C, which predicts transitions between the concepts. We represent this function by a set of functions, one per concept (in this case: F_s, F_u, F_h). In (g)-(i) we approximate the output behaviour of the RNN by a function S, which predicts the output class from a concept. This function is represented by a concept-class mapping, specifying an output label for every concept (in this case: *healthy* $\rightarrow$ 0, *sick* $\rightarrow$ 1, and *unc* $\rightarrow$ 1). Collectively, steps (1)-(3) are used to produce our extracted model, consisting of concepts, their interactions, and their corresponding class labels.

the model outputs predicted labels $y \in \{0, 1\}$. Furthermore, we assume that we can access/generate predicted labels at all timesteps, even if the task itself is defined as a whole-sequence prediction task.

3.2 Concept Extraction

The first step of our approach approximates the hidden space of a given RNN with a set of concepts C, as shown in Figure 1 (a)-(c).

Similarly to existing concept extraction approaches (such as [16]), we extract concepts by obtaining summaries of clusters in the hidden space. We use f_{hidden} to compute the hidden representation of the RNN's training data points, and cluster it using *Kmeans* clustering to generate the set of concepts C, such that every cluster is associated with a concept in C.

Unlike existing work on concept extraction, which focuses predominantly on image-based tasks, we focus on multivariate, continuous, temporal data tasks. Consequently, we argue that explaining the extracted clusters via prototypical examples is not suitable in our context, and that higher-level explanations should be provided instead. In this work, we rely on the class label space when generating concepts, by introducing a majority-voting cluster naming strategy we refer to as the *majority labelling* approach, which will be described in the remainder of this section.

Given a cluster k consisting of p data-points in their hidden representation $(\mathbf{h}_1, ..., \mathbf{h}_p)$, and the corresponding class labels of these data-points $(y_1, ..., y_p)$, we define the *majority label* (MaL) and

the *majority label ratio* ($MaLR$) for that cluster as the most frequently-occurring class label, and it's corresponding ratio of occurrence, respectively, as shown in Equation 1 and Equation 2. In Equation 1, L refers to the class label space (thus, in this paper $L = \{0, 1\}$).

$$MaL_k = \arg\max_{l \in L} \sum_{i=1}^{p} [y_i = l] \tag{1}$$

$$MaLR_k = \frac{1}{p} \sum_{i=1}^{p} [y_i = MaL_k] \tag{2}$$

Given MaL_k and $MaLR_k$ for a cluster k, we define it's associated concept c_k as shown in Equation 3, where $ClassName(MaL_k)$ refers to the class name associated with the class label.

$$c_k = ClassName(MaL_k) \ \textbf{ if } MaLR_k > \theta \textbf{ ,} \ unc. \textbf{ otherwise} \tag{3}$$

Hence, the concept information associated with a cluster is the name of the most-frequently occurring class in that cluster, or *uncertain* (written as *unc.* for short), if the $MaLR$ is less than a predefined threshold θ.

3.3 Transition Function Extraction

The second step of our approach approximates the RNN's transitions between the extracted concepts, as shown in Figure 1 (d)-(f). We achieve this by approximating the RNN's hidden function f_{hidden} with the function $F_C : \mathbb{R}^n \times C \rightarrow C$ representing the RNN's transition dynamics between concepts. In practice, F_C needs to be interpretable (in order to facilitate consequent interpretation of the extracted model), whilst faithfully approximating the original function f_{hidden} (to ensure high fidelity). To meet these criteria, we define F_C as a set of interpretable functions, one per concept: $F_C = \{F_c \mid c \in C\}$. Each function $F_c : \mathbb{R}^n \rightarrow C$ captures the transition dynamics for a concept c. That is, given an input $\mathbf{x}_{t-1}$, $F_c(\mathbf{x}_{t-1})$ predicts the concept the RNN will be in at time t, assuming the RNN was in concept c at time $t-1$. Crucially, the overall complexity of f_{hidden} is now decomposed into several simpler functions, which can be represented by less complex, more interpretable functions, such as Decision Trees (DTs).

We treat the functions in F_C as classifiers, mapping an input $\mathbf{x} \in \mathbb{R}^n$ to a concept $c \in C$. Consequently, for training, these functions are setup as multi-label classification problems, where the functions are equipped with labelled data consisting of input points $\mathbf{x}$, and corresponding concept labels C. We obtain the transition function training data directly from the original RNN training data. Further details regarding MEME transition functions can be found in Appendix B.1 and Appendix B.2.

3.4 MEME Extracted Models

The third model extraction step consists of producing a mapping from concepts to output classes, which we refer to as $S : C \rightarrow L$, as shown in Figure 1 (g)-(i). We define the output class corresponding to a concept as the majority label of its corresponding cluster MaL.

Finally, we define models extracted by MEME as a tuple (C, F_C, S, c_0). We assume that the RNN has a fixed initial hidden state $\mathbf{h}_0$, which is located in a cluster corresponding to c_0 (i.e., the starting concept). Similarly to an FSA, a MEME model processes an input sequence by performing a series of state transitions, such that at timestep t, the model computes the next state from the current state and the next input sample, and outputs a class label based on this next state. However, unlike FSA-based approaches, MEME models represent states by interpretable concepts C, and represent transition functions by classifiers in F_C, capable of processing multivariate, continuous input. The sequence classification process of MEME models is summarised in Figure 2. The sequence classification algorithm used by MEME models is given in Appendix B.3.

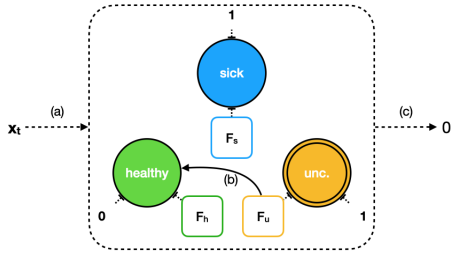


Figure 2: MEME model input processing example. The concept c_{t-1} which the model is in at time $t-1$ is highlighted with a double border. (a) At time t, the model receives the next input $\mathbf{x}_t$. (b) After receiving the input, the model uses $F_{c_{t-1}}$ (in this case: F_u) to compute the next concept c_t (in this case: *healthy*), from the current concept c_{t-1} (in this case: *unc*) and the new input $\mathbf{x}_t$. (c) The model outputs the label corresponding to the next concept c_t (in this case: 0), as the output class label of the input $\mathbf{x}_t$.

4 Experiments

We evaluated our approach using two case-studies: Room Occupation Prediction (ROP), and In-Hospital Mortality Prediction (IHMP). When extracting concepts, we used the RNN recurrent layers closest to the output as the layers computing f_{hidden}, for both case-studies. We experimented with using Decision Trees (DTs), and Multi-Layer Perceptrons (MLPs) as transition function types. Further details regarding extracted model specifications can be found in Appendix C. Relevant details and code used to run the experiments can be found in our repository.[3]

Room Occupation Prediction The ROP task focuses on predicting room occupancy using measurements of environmental variables. For this task, we make use of the *Occupancy Detection Data Set* available from the UCI Machine Learning repository.[4] This is a binary classification problem in which observations of sensor readings (such as Temperature, Humidity, or Light) are used to determine whether a room is occupied or unoccupied [4]. Further details can be found in Appendix D.

In-Hospital Mortality Prediction For our second case-study, we used the *in-hospital mortality prediction* task defined in [3]. This work presents a public repository[5] with 4 clinical benchmark tasks (including IHMP), all derived from the Medical Information Mart for Intensive Care (MIMIC-III) dataset [2]. The IHMP task focuses on predicting patient in-hospital mortality based on the first 48 hours of a patient's ICU stay, using a set of clinical variables monitored over time. We used the data extraction pipeline described in [3] for producing the IHMP task dataset from the MIMIC-III data. Further details can be found in Appendix E.

5 Results

This section presents the results obtained by evaluating our approach using the two case studies described above. Section 5.1 shows the extracted concepts, and the effect of varying concept granularity. Section 5.2 performs global inspection of the transition functions. Section 5.3 demonstrates how the extracted models perform sequence prediction, offering local explanations. Section 5.4 measures the fidelity of the extracted models, and their performance on the original tasks. Further experimental results can be found in Appendix F.

[3] Code will be released after paper review.

[4] https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

[5] https://github.com/YerevaNN/mimic3-benchmarks/

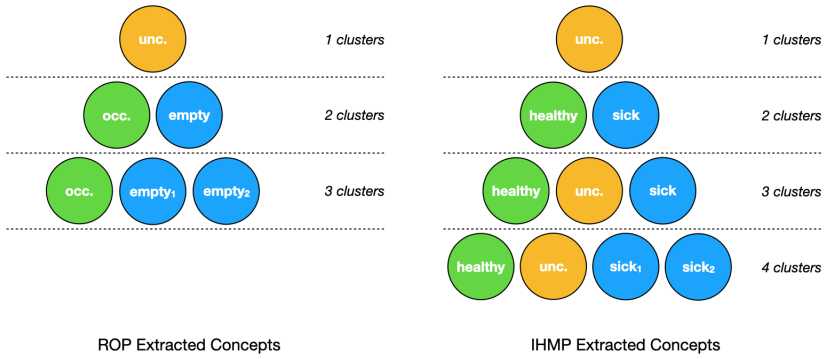


Figure 3: The concepts extracted from the RNNs at different clustering levels. Larger cluster numbers exhibited even more repeating concepts than those presented here, and are thus not shown.

5.1 Concept Granularity

We inspect the extracted concepts, and observe how the number of *Kmeans* clusters affects their granularity. Figure 3 shows the extracted concepts obtained at varying numbers of clusters.

A coarse clustering of one cluster yields the *uncertain* concept for both the ROP and the IHMP tasks, since a single cluster consists of a mixture of points from both positive and negative classes. At two clusters, the concepts correspond to the positive and negative class labels. This is expected, since linear separability of the hidden space of a DL model w.r.t. its output classes increases with depth [34], and is high in layers close to the output. At 3 clusters, the IHMP task produces an extra *uncertain* concept, indicating that the RNN hidden space contains a sub-region corresponding to states in which the RNN is uncertain. At higher concept granularity (4 clusters for IHMP, and 3 clusters for ROP), the clustering step begins to return concepts represented by the same concept labels. Given the rich amount of information contained in the RNN hidden space, these more fine-grained clusterings likely hold information on intra-class concepts (e.g., in case of the IHMP task, $sick_1$ and $sick_2$ may correspond to different types of sickness progression).

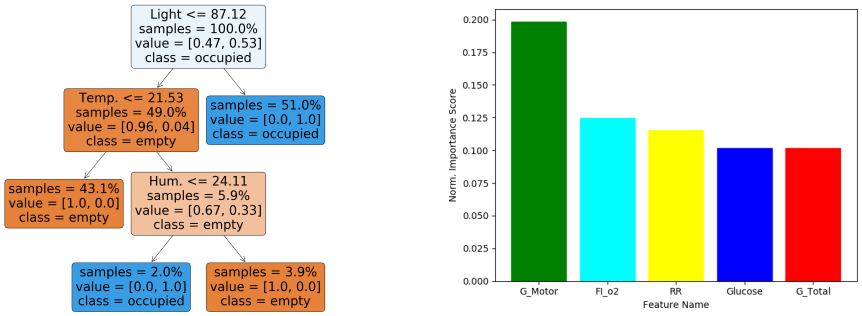
Overall, for a small cluster number, our *majority labelling* approach offers a straightforward way of interpreting the clusters via their relation to the output classes. For the rest of the experiments in this section, we fix the number of clusters to those producing the most granular concepts with distinct names: 2 clusters in the case of ROP, and 3 in the case of IHMP.

5.2 Transition Function Interpretation

In this section, we inspect the transition functions of the extracted models, showing how they can be used to better understand global behaviour and concept interplay of the extracted models. For the sake of space, we focus on a representative subset of the extracted model transition functions, instead of presenting all of them.

Since DT transition functions are interpretable, we can inspect them directly. Figure 4(a) shows the DT transition function for the *occupied* concept of the ROP task. Using this plot, it is possible to directly interpret the reasons for the transition function predicting a transition to the *occupied* concept, or to the *empty* concept. In this example, *Light* is the most important feature, with low light indicating that the room is empty. This is likely due to the fact that the first action a room occupant undertakes is to switch on the lights (further details regarding the sensors, room, etc. can be found in [4]). The remaining features exhibit similar intuitive justifications.

In cases where transition functions cannot be interpreted directly (e.g., MLPs), inspection can be done by relying on existing XAI approaches. The field of XAI offers a wide variety of approaches targeted at global explanations of MLPs, including (but not limited to): Partial Dependence Plots (PDPs) [11], model distillation [5], SP-LIME [28], GENN [6], and many others [11]. The results of one such approach are shown in Figure 4(b), in which we plot the top 5 most important features of the MLP transition function for the *uncertain* concept, using the *permutation feature importance*



(a) (b)

Figure 4: Transition function interpretation examples. (a) DT transition function for the *occupied* concept of the ROP task. (b) Top 5 most important features and their normalised importance scores for the *uncertain* concept MLP transition function of the IHMP task.

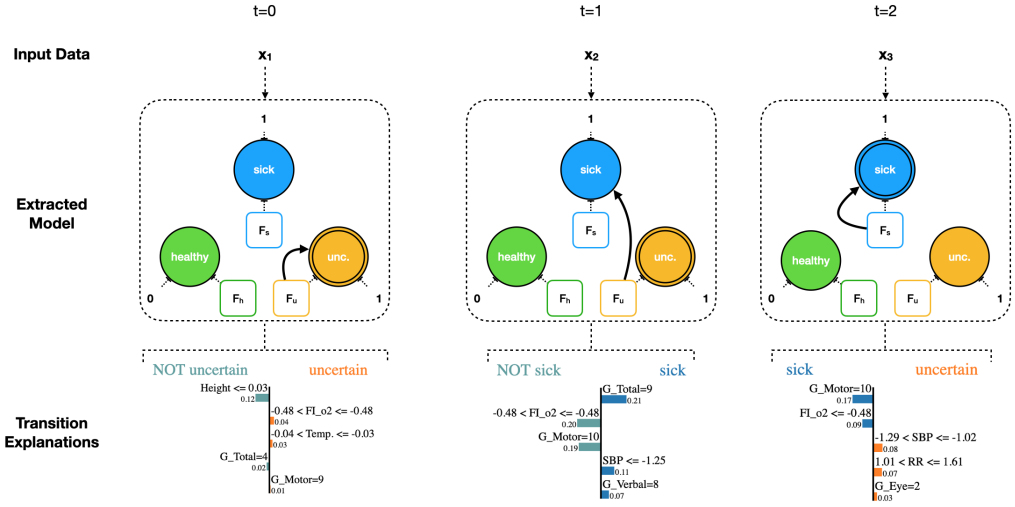


Figure 5: Extracted model sequence processing for three timesteps ($t = 0, 1, 2$), with *uncertain* as the initial concept. For each timestep t, the concept the model is in at time t is highlighted with a double border. We show the input data ($\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$), the corresponding concept transition sequence (*uncertain* $\rightarrow$ *uncertain* $\rightarrow$ *sick* $\rightarrow$ *sick*), and the explanations for each transition function prediction. In this example, the class labels outputted by the model are not shown.

scoring approach [29]. Consequently, we are able to better understand the transition behaviour of the MLP function of that concept, and the features it mostly relies on.

5.3 Local Interpretability

In addition to global interpretability, we can obtain local interpretability of an RNN's individual instance prediction by observing sequence processing of the extracted model. Due to how our extracted models are defined, any label produced by such a model at any timestep can be directly traced back to its concepts and their transitions. In case of interpretable transition functions, such as DTs, transition function predictions can be explained directly. In case of black-box transition functions, such as MLPs, we can employ existing methods for producing local explanations of black-box classifier predictions [11]. For instance, Figure 5 shows how the MLP-based extracted model of the IHMP task processes a sequence of 3 timesteps, outputting local explanations of the MLP transition function predictions for each timestep. Here, we use the LIME approach [28] (tabular explainer with discretised continuous values).

Table 1: Accuracy Comparison

		ROP	IHMP
Original Model		96.1 +/- 0.1%	85.5 +/- 1.1%
Extracted Models	DT	95.8 +/- 0.3%	79.7 +/- 1.6%
	MLP	96.0 +/- 0.3%	82.3 +/- 0.9%

Table 2: Approximation Quality Comparison

		ROP	IHMP
Fidelity	DT	95.9 +/- 0.3%	87.5 +/- 1.0%
	MLP	97.1 +/- 0.3%	91.7 +/- 0.8%
Precision	DT	0.92 +/- 0.02	0.96 +/- 0.01
	MLP	0.93 +/- 0.01	0.92 +/- 0.01
Recall	DT	0.98 +/- 0.03	0.81 +/- 0.02
	MLP	0.99 +/- 0.01	0.93 +/- 0.01
F1	DT	0.95 +/- 0.01	0.88 +/- 0.01
	MLP	0.96 +/- 0.01	0.92 +/- 0.01

5.4 Model Performance

The utility of our extracted models was also evaluated quantitatively by measuring their performance on the original classification tasks, and their closeness of approximation of the original models. The predictive accuracy of the original RNNs and the extracted models is shown in Table 1. For the simpler ROP task, both extracted model types achieved performance close to that of the original model. For the relatively more complex IHMP prediction task, extracted models exhibited a slight reduction in performance. However, this reduction was smaller for the MLP-based model, which is likely a consequence of MLPs being more powerful models than DTs [33].

Table 2 shows the fidelity, precision, recall, and F1 scores of the extracted models, computed by treating the original RNN predictions as the ground truth. Overall, both extracted model types achieved high approximation scores on both tasks. Similarly to the results in Table 1, the MLP-based extracted models achieved a higher overall approximation (fidelity and F1 scores), than the DT-based extracted models, with the difference being more pronounced in the case of the more challenging IHMP task.

These results demonstrate that our approach generates models which are faithful approximations of the original RNNs. Furthermore, they show that for more complex tasks, using more powerful transition functions leads to better performance.

6 Conclusions

We presented MEME, a model extraction approach capable of approximating RNNs with interpretable models represented by human-understandable concepts and their interactions. We evaluated our approach using two multivariate, continuous data case studies (including a realistic healthcare scenario). Using these case-studies, we demonstrated that extracted models produced by MEME faithfully approximate their original RNN models, and can be used to extract useful knowledge from them.

MEME is a highly modular approach which can be extended in a variety of ways. Firstly, MEME can be combined with other transition function types, such Random Forests, or Dynamic Time-Warping models [19]. Secondly, the majority labelling concept extraction approach can be extended by relying on other types of information besides class labels, such as sample *sequence motifs* [1], or cluster descriptor tag assignments [36]. Exploring the effects these approaches have on fidelity, concept granularity, concept representation, etc. is an interesting avenue for further exploration.

Overall, given the rapidly increasing interest in time-series DL systems (in the field of healthcare, in particular), we believe our approach can play an important role in enriching such systems with explainability and knowledge extraction capabilities.

References

[1] Yifeng Gao and Jessica Lin, "Discovering Subdimensional Motifs of Different Lengths in Large-Scale Multivariate Time Series", ICDM, 2019

[2] Johnson, Alistair EW and Pollard, Tom J and Shen, Lu and Li-wei, H Lehman and Feng, Mengling and Ghassemi, Mohammad and Moody, Benjamin and Szolovits, Peter and Celi, Leo Anthony and Mark, Roger G, "MIMIC-III, a freely accessible critical care database", Scientific data, vol. 3, 2016

[3] Harutyunyan, Hrayr and Khachatrian, Hrant and Kale, David C. and Ver Steeg, Greg and Galstyan, Aram, "Multitask learning and benchmarking with clinical time series data", Scientific Data, vol. 6, 2019

[4] Luis M. Candanedo and Veronique Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models", Energy and Buildings, vol. 112, 2016

[5] Sarah Tan and Rich Caruana and Giles Hooker and Paul Koch and Albert Gordo, "Learning Global Additive Explanations for Neural Nets Using Model Distillation", arXiv 1801.08640, 2018

[6] Mark Ibrahim, Melissa Louie, Ceena Modarres, and John Paisley, "Global Explanations of Neural Networks: Mapping the Landscape of Predictions", AAAI/ACM Conference on AI, Ethics, and Society, 2019

[7] Tarwani, Kanchan M and Edem, Swathi, "Survey on recurrent neural network in natural language processing", Int. J. Eng. Trends Technol, vol. 48, 2017

[8] Miotto, Riccardo and Wang, Fei and Wang, Shuang and Jiang, Xiaoqian and Dudley, Joel T, "Deep learning for healthcare: review, opportunities and challenges", Briefings in bioinformatics, vol. 19, 2018

[9] Zhang, Qingchen and Yang, Laurence T and Chen, Zhikui and Li, Peng, "A survey on deep learning for big data", Information Fusion, vol. 42, 2018

[10] Gao, Lianli and Guo, Zhao and Zhang, Hanwang and Xu, Xing and Shen, Heng Tao, "Video captioning with attention-based LSTM and semantic consistency", IEEE Transactions on Multimedia, vol. 19, 2017

[11] Molnar, Christoph, "Interpretable machine learning", 2019

[12] Bastani, Osbert and Kim, Carolyn and Bastani, Hamsa, "Interpreting blackbox models via model extraction", arXiv:1705.08504, 2017

[13] Zeiler, Matthew D and Fergus, Rob, "Visualizing and understanding convolutional networks", European conference on computer vision, 2014

[14] Adebayo, Julius and Gilmer, Justin and Muelly, Michael and Goodfellow, Ian and Hardt, Moritz and Kim, Been, "Sanity checks for saliency maps", NeurIPS, 2018

[15] Dimanov, Botty and Bhatt, Umang and Jamnik, Mateja and Weller, Adrian, "You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods", 2020

[16] Ghorbani, Amirata and Wexler, James and Zou, James Y and Kim, Been, "Towards automatic concept-based explanations", NeurIPS, 2019

[17] Kim, Been and Wattenberg, Martin and Gilmer, Justin and Cai, Carrie and Wexler, James and Viegas, Fernanda and Sayres, Rory, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)", arXiv:1711.11279, 2017

[18] Goyal, Yash and Shalit, Uri and Kim, Been, "Explaining Classifiers with Causal Concept Effect (CaCE)", arXiv:1907.07165, 2019

[19] Xing, Zhengzheng and Pei, Jian and Keogh, Eamonn, "A brief survey on sequence classification", ACM Sigkdd Explorations Newsletter, vol. 12, 2010

[20] Weiss, Gail and Goldberg, Yoav and Yahav, Eran, "Learning Deterministic Weighted Automata with Queries and Counterexamples", NeurIPS, 2019

[21] Hou, Bo-Jian and Zhou, Zhi-Hua, "Learning With Interpretable Structure From Gated RNN", IEEE, 2020

[22] Du, Xiaoning and Xie, Xiaofei and Li, Yi and Ma, Lei and Liu, Yang and Zhao, Jianjun, "Deepstellar: model-based quantitative analysis of stateful deep learning systems", ACM, 2019

[23] Choi, Edward and Bahadori, Mohammad Taha and Sun, Jimeng and Kulas, Joshua and Schuetz, Andy and Stewart, Walter, "Retain: An interpretable predictive model for healthcare using reverse time attention mechanism", NeurIPS, 2016

[24] Bai, Tian and Zhang, Shanshan and Egleston, Brian L and Vucetic, Slobodan, "Interpretable representation learning for healthcare via capturing disease progression through time", ACM SIGKDD, 2018

[25] Wu, Mike and Hughes, Michael C and Parbhoo, Sonali and Zazzi, Maurizio and Roth, Volker and Doshi-Velez, Finale, "Beyond sparsity: Tree regularization of deep models for interpretability", AAAI, 2018

[26] Sarah Wiegreffe and Yuval Pinter, "Attention is not not Explanation", EMNLP-IJCNLP, 2019

[27] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research, vol. 12, 2011

[28] Ribeiro, Marco Tulio and Singh, Sameer and Guestrin, Carlos, "Why should i trust you?" Explaining the predictions of any classifier, ACM SIGKDD, 2016

[29] Altmann, André and Toloşi, Laura and Sander, Oliver and Lengauer, Thomas, "Permutation importance: a corrected feature importance measure", Bioinformatics, vol. 26, 2010

[30] Fawaz, Hassan Ismail and Forestier, Germain and Weber, Jonathan and Idoumghar, Lhassane and Muller, Pierre-Alain, "Deep learning for time series classification: a review", Data Mining and Knowledge Discovery, vol. 33, 2019

[31] Hinton, Geoffrey E, "Learning multiple layers of representation", Trends in cognitive sciences, vol. 11, 2007

[32] Bolei Zhou and Aditya Khosla and Àgata Lapedriza and Aude Oliva and Antonio Torralba, "Object Detectors Emerge in Deep Scene CNNs", ICLR, 2015

[33] Bengio, Yoshua and Delalleau, Olivier and Simard, Clarence, "Decision trees do not generalize to new variations", Computational Intelligence, vol. 26, 2010

[34] Alain, Guillaume and Bengio, Yoshua, "Understanding intermediate layers using linear classifier probes", arXiv:1610.01644, 2016

[35] Craven, Mark and Shavlik, Jude W, "Extracting tree-structured representations of trained networks", NeurIPS, 1996

[36] Sambaturu, Prathyush and Gupta, Aparna and Davidson, Ian and Ravi, SS and Vullikanti, Anil, and Warren, Andrew, "Efficient Algorithms for Generating Provably Near-Optimal Cluster Descriptors for Explainability", AAAI, 2020

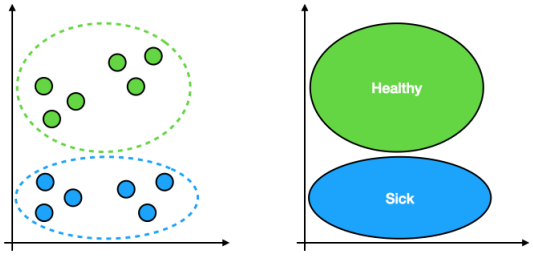
[37] Chen, Daizhuo and Fraiberger, Samuel P and Moakler, Robert and Provost, Foster, "Enhancing transparency and control when drawing data-driven inferences about individuals", Big data, 2017

[38] Sato, Makoto and Tsukimoto, Hiroshi, "Rule extraction from neural networks via decision tree induction", IJCNN, 2001

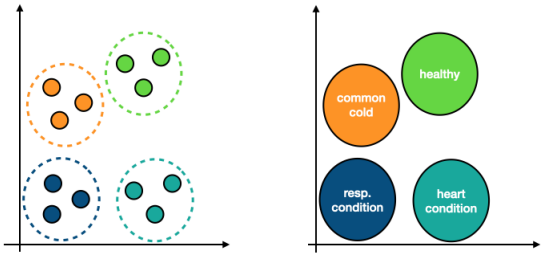
[39] Kazhdan, Dmitry and Shams, Zohreh and Liò, Pietro, "MARLeME: A Multi-Agent Reinforcement Learning Model Extraction Library", arXiv:2004.07928, 2020

A Concept-based Explanations Further Details

Concept-based explanations are explanations consisting of high-level, human-understandable "concepts". In practice, giving a precise definition for what constitutes a concept from a psychological and/or philosophical standpoint is highly non-trivial [16]. Instead, in line with recent work, we refer to a concept as a global explanation of the internal workings of a model, presented in higher-level, human-understandable units, rather than individual features, pixels, or characters. Concept-based explanations are intended to serve as more understandable, usable explanations of DL models, and



(a) Low Granularity



(b) High Granularity

Figure 6: Anticipated effect of cluster number on concept granularity. The subfigures on the left demonstrate two different *Kmeans* clusterings of the same hidden representation. The subfigures on the right demonstrate the corresponding extracted concepts. (a) A small cluster number yields coarse concepts (*sick* and *healthy*). (b) A higher cluster number increases concept granularity. In this case, extracted concepts now represent specific sicknesses/conditions (*common cold*, *respiratory condition* and *heart condition*).

have been used in a range of different ways, including: inspecting what the model has learned [16], providing class-specific explanations [17], and discovering causal relations of concepts in data [18]. Most recent work on concept extraction (extracting learned concepts from DL models) focuses on image-recognition tasks, representing concepts by images (e.g., concepts associated with the class "dog" may be images of paws, ears, or tails) [17].

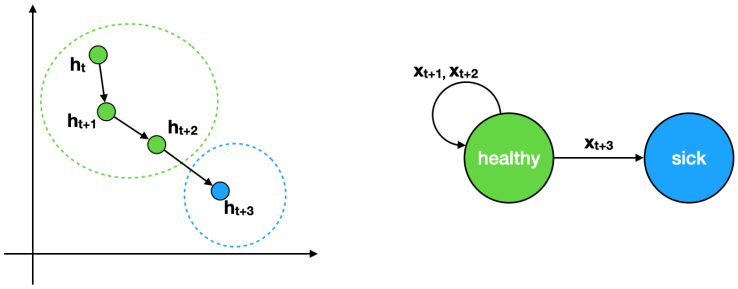
Deep Neural Networks (DNNs) have been shown to perform hierarchical feature extraction, with layers closer to the output utilising higher-level data representations, compared to layers closer to the input [31, 32]. Consequently, recent work on concept extraction relies on euclidean distance in the activation space of final layers (referred to as the *hidden space*) of DNN models as an effective concept similarity metric [16]. The intuition is that data-points exhibiting similar high-level features will be clustered together in the hidden space of the corresponding DNN layers, implying that the hidden space is expected to demonstrate conceptual grouping. Thus, one approach to concept extraction is to cluster data points in the activation space of higher-level model layers, and extracting cluster summaries in the form of prototypical samples, to serve as concept representations [16].

Intuitively, we expect the number of clusters to affect the granularity of the corresponding extracted concepts. Thus, a low cluster number will likely correspond to extraction of coarse concepts, and a high cluster number will likely increase concept granularity. An example is given in Figure 6.

B MEME Model Extraction Further Details

B.1 Transition Training Data Extraction

For every time-series sequence $X_s = (\mathbf{x}_1, ..., \mathbf{x}_T)$ in the RNN training data, we obtain the corresponding RNN hidden states $H_s = (\mathbf{h}_1, ..., \mathbf{h}_T)$, and use H_s to obtain the corresponding concept sequence $C_s = (c_1, ..., c_T)$. We then use C_s to produce the sample transition dataset $Tr_s = ([c_0, \mathbf{x}_1, c_1], ..., [c_{T-1}, \mathbf{x}_T, c_T])$, such that every item in Tr_s consists of the concept at time $t-1$, the input point at time t, and the next concept at time t, for t ranging from 1 to T. We aggregate these transition samples Tr_s over all training sequences, to produce the overall transition dataset Tr.



(a) Hidden Space (b) Quantised Space

Figure 7: (a) When processing an input sequence ($\mathbf{x}_{t+1}$,$\mathbf{x}_{t+2}$,$\mathbf{x}_{t+3}$), the RNN outputs a corresponding sequence of points ($\mathbf{h}_{t+1}$,$\mathbf{h}_{t+2}$,$\mathbf{h}_{t+3}$) in the hidden space (assumed to be two-dimensional in this example). These points move within a cluster (intra-cluster dynamics), before moving into another cluster (inter-cluster dynamics). (b) Due to quantisation, the intra-cluster movement of datapoints is not observable, implying that the extracted model only observes the final sample ($\mathbf{x}_{t+3}$) causing the model to transition to the other cluster. Note: in (b), we omit the transition functions and concept-class map here for clarity, showing only the concept sequence.

Using Tr, the training datasets D_c for every transition function F_c are defined as:

$$D_c = \{(\mathbf{x}_t, c_t) \mid [c_{t-1}, \mathbf{x}_t, c_t] \in Tr \ \wedge \ c_{t-1} = c\} \tag{4}$$

In practice, the nature and sizes of the transition datasets D_c are unpredictable. Hence, their important properties (e.g., class imbalance, size, or representativity) are difficult to anticipate/control. For now, we account for one of the mentioned issues (class imbalance) by balancing out every dataset D_c (via downsampling), ensuring that they contain an equal number of class samples.

B.2 Transition Function Context Window

The clustering quantisation step described in Section 3.2 discards information about the *intra-cluster dynamics* of an RNN (i.e., hidden space dynamics within clusters), and retains only *inter-cluster dynamics* information, as summarised in Figure 7. In practice, this is a potential source of approximation error, which can adversely affect extracted model fidelity. We address this issue by introducing a *context window* hyper-parameter w to our transition functions, allowing them to retain a context window containing w previous input points. In this case, the transition functions are defined as $F_c : \mathbb{R}^{n*(w+1)} \rightarrow C$, where F_c takes the current input $\mathbf{x}_t$, together with w previous inputs $\mathbf{x}_{t-1}, ..., \mathbf{x}_{t-w}$ concatenated together, when predicting the next concept c_t.

B.3 Sequence Processing

Algorithm 1 classifySequence($sequence$)

```
c = c_0
labels = []
for t ∈ [w, ..., T] do
    label = S(c)                          ▷ Get next output label
    labels.append(label)
    input = sequence[(t − w) : t]
    c = F_c(input)                        ▷ Get next concept
end for
return labels
```

In Algorithm 1, w is the transition function context window size (assumed to be less than the sequence length T), $sequence$ is an input sequence to be classified, and F_c denotes the transition function of a concept c. In case of full-sequence classification (classifying not every timestep, but the entire sequence), we simply retrieve the last label from $labels$. We have also implemented a more efficient,

batched implementation of Algorithm 1 (not shown here for the sake of space), which can be found in our repository.

C Experimental Setup Further Details

C.1 Concept Extraction

When selecting a value of the threshold θ used for computing concept names, we found that setting $\theta = 0.8$ works well in practice. Thus, all results reported in Section 5 were computed with θ fixed to this value.

C.2 Transition Functions

For DTs, we relied on the *DTClassifier* implementation provided in scikit-learn [27]. Restricting the maximum depth of the DTs exhibited a beneficial regularising effect, reduced overfitting and improving predictive performance, whilst also making inspection easier. Setting a maximum depth of 2 worked best for the ROP task, and a depth of 3 for the IHMP task.

For MLPs, we used a 3-layer MLP, consisting of a softmax output layer, and two hidden ReLu dense layers of sizes 200 and 50. We used the *Keras* API available in *Tensorflow*[6] for setting up the models, and trained with the standard categorical crossentropy loss. Further details can be found in the repository.

D ROP Task Further Details

D.1 Features & Preprocessing

The original dataset consists of 5 features: *Temperature* (Temp.), *Humidity* (Hum.), *Light*, *Humidity Ratio*, and *Co2*. We found that the *Humidity Ratio* feature was practically equivalent to the *Humidity* feature (up to a scale factor), and therefore discarded it. Hence, our ROP dataset consisted of 4 features.

The original dataset consists of three sequences of sensor readings taken at regular 60s intervals (i.e., the train/test/val datasets each consist of a single long sequence). In order to produce multi-sample datasets, we split each of these 3 sequences into subsequences of 60 timesteps each.

D.2 RNN Model

The RNN model trained for this task consisted of 2 hidden LSTM layers of sizes 100 and 50, with dropout layers in-between, and a sigmoid output layer. The model was trained with a binary crossentropy loss.

D.3 Classes

The problem is formulated as a binary classification problem, with class 0 corresponding to an empty room (referred to as *empty* in the rest of this work), and class 1 to an occupied room (referred to as *occupied* in the rest of this work, and abbreviated as *occ.*). The RNN was trained to compute a predicted label at every timestep.

E IHMP Task Further Details

E.1 Features & Preprocessing

For our RNN model, we relied on the data representation used in [3]. The input features consist of 17 clinical variables collected during the first 48 hours of a patient's stay. Table 3 lists the input features, together with their corresponding acronyms, and whether they are continuous or not (referred to as *type*). In addition to the 17 *clinical* variable features, the feature set also included 17 binary features

[6] https://www.tensorflow.org/guide/keras/overview

Table 3: IHMP Feature Names, Acronyms, and Types

Name	Acronym	Type
Capillary refill rate	CRR	Categorical
Glascow coma scale eye opening	G_Eye	Categorical
Glascow coma scale motor response	G_Motor	Categorical
Glascow coma scale total	G_Total	Categorical
Glascow coma scale verbal response	G_Verbal	Categorical
Diastolic blood pressure	DBP	Continuous
Fraction inspired oxygen	FI_o2	Continuous
Glucose	Glucose	Continuous
Heart Rate	HR	Continuous
Height	Height	Continuous
Mean blood pressure	MBP	Continuous
Oxygen saturation	o2S	Continuous
Respiratory rate	RR	Continuous
Systolic blood pressure	SBP	Continuous
Temperature	Temp.	Continuous
Weight	Weight	Continuous
pH	pH	Continuous

(one per clinical feature), specifying whether the values of the clinical variables were observed, or inferred (these binary features are referred to as the *non-clinical* features). We used this original feature representation without modification when training the RNN model.

For the extracted models, we relied on a slightly modified input representation: the categorical features were represented in their categorical format (as opposed to the one-hot encoded format used for the RNN), and the *non-clinical* features were omitted. This configuration gave slightly better extracted model performance, and also made the extracted models easier to interpret.

Before training the RNN, we balanced the dataset via down-sampling, ensuring an equal number of positive and negative class samples. Furthermore, we focused on extracting models encapsulating the RNN's typical behaviour, instead of edge-case behaviour, by only using data the RNN was more confident in. This was achieved by using the RNN's sigmoid output as a proxy for confidence, and selecting "high-confidence" points from the balanced dataset, for which this output was either in $[0, 0.2]$, or in $[0.8, 1.0]$ (i.e., "closer to 0", or "closer to 1"). Extracted model training and evaluation was consequently performed using this "high-confidence" data.

E.2 Model

The RNN model trained for this task consisted of 3 hidden LSTM layers of sizes 256, 100 and 100, with dropout layers in-between, and a sigmoid output layer. The model was trained with a binary crossentropy loss.

E.3 Classes

As discussed previously in Section 4, the task is formulated as a binary classification problem, with class 1 corresponding to a patient dying in the first 48 hours of ICU stay (referred to as *sick*), and class 0 to a patient not dying (referred to as *healthy*). The RNN was trained for sequence prediction, computing a predicted label only at the end of an entire sequence.

F Further Results

F.1 Transition Function Interpretation

Similarly to the ROP task, Figure 8 shows a DT transition function of the IHMP task. As in the ROP example, these plots can be used to inspect the reasons for the extracted model concept transitions, which could be used to better understand the nature of sickness progression and the key reasons for the model to "change it's mind" over time.

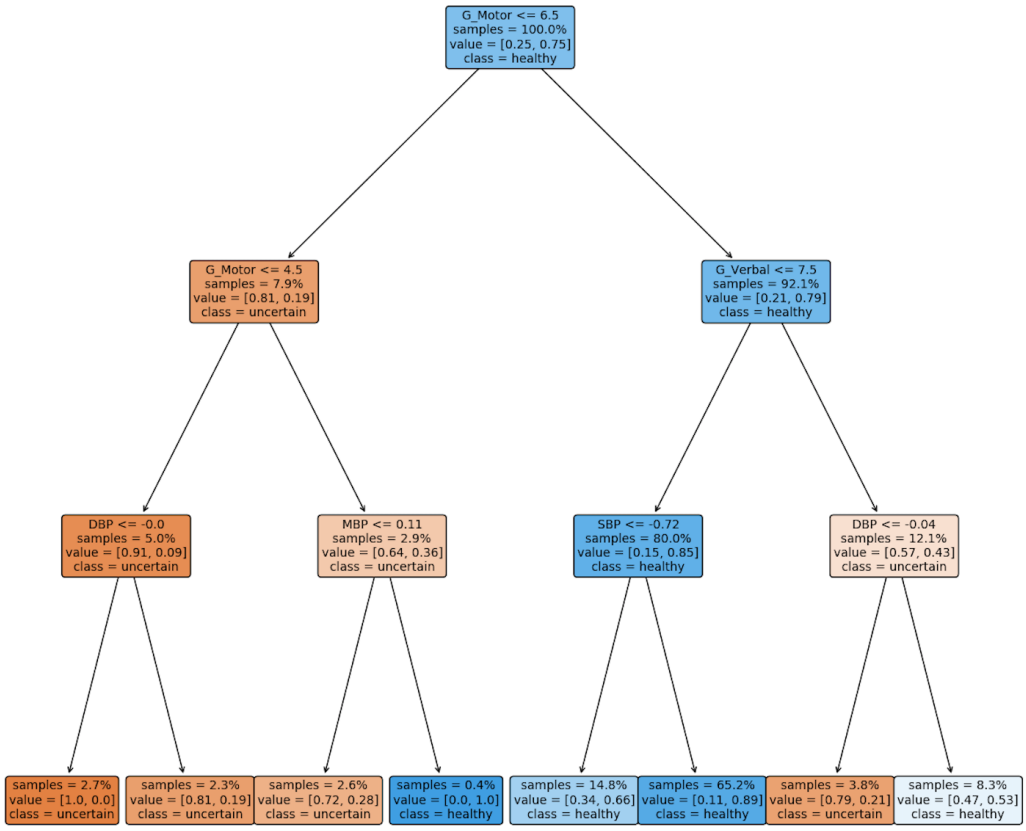
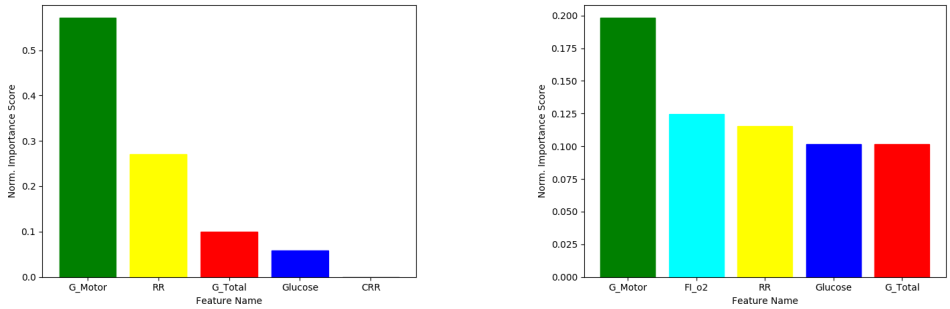


Figure 8: DT transition function for the *healthy* concept of the IHMP task. The original RNN model never exhibited a direct transition from the *healthy* concept to the *sick* concept. Thus, this DT only has two output classes (*uncertain* and *healthy*).



(a) (b)

Figure 9: Top 5 most important features and their normalised importance scores for the *uncertain* concept of the DT(a), and MLP(b) extracted models for the IHMP task.

F.2 Transition Function Comparison

Figure 9 shows the top 5 most important feature scores of the transition function for the *uncertain* concept, for both the DT and MLP extracted models. The top 4 features of the DT model are all in the top 5 for the corresponding MLP model, indicating that both models were capable of identifying and using the discriminative power of the same features.

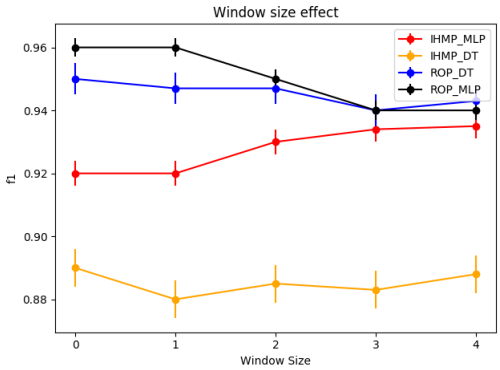


Figure 10: Effect of window size w on the F1 score.

Interestingly, the MLP model has the *FI_O2* feature as the second most important one, which might be due to the MLP capability of relying on more complex, non-linear features, unusable by a DT [33]. More generally, Figure 9 demonstrates that it is possible to compare and contrast the results of different transition function types in order to better understand the underlying features and model behaviour.

F.3 Context Window Size

For the experiments described in Section 5, Appendix F.1, and Appendix F.2, we set a context window size of $w = 0$ (i.e., the transition functions retain no context). In this section, we quantitatively evaluated the effect of using the transition function context window described in Appendix B.2. Figure 10 shows the effect of the context window size w on the $F1$ approximation score of the extracted models.

For the ROP task, both the DT and MLP models exhibited a $\sim$1-2% decrease in performance with increasing window size. The likely reason for this is that the input features at the current timestep are sufficient for good performance in case of simpler tasks, such as ROP. In such cases, a context window introduces extra unnecessary input features to the transition functions, harming their performance, and the overall performance of the extracted model.

For the IHMP task, increasing the window size did not affect the DT-based model score significantly, but resulted in a $\sim$2% increase in performance for the MLP-based model. The likely reason for this is that, unlike the MLP transition functions, the DT transition functions were not able to extract information signals contained in the context information [33].

Overall, our results indicate that for complex tasks (such as IHMP) and for powerful transition functions (such as MLPs), a context window may improve extracted model approximation quality. However, this might not be the case for relatively simpler tasks (such as ROP) or simpler transition functions (such as DTs). In future work, we intend to explore context window size variation in more detail.