Extracting finite state representations from recurrent models of Industrial Cyber Physical Systems

Alessandro Brusaferri[a,b], Matteo Matteucci[b], Stefano Spinelli[a,b], Andrea Vitali[a]
[a]*CNR-Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing, Milan, Italy*
[b]*Politecnico di Milano, Milan, Italy*
$name.surname$@stiima.cnr.it, $name.surname$@polimi.it

***Abstract*—Neural networks are being broadly explored for the identification of Industrial Cyber Physical Systems (ICPS) models from data sequences. However, learned representations typically lack explainability, representing nowadays a major challenge of deep learning. Interpreting the information structured across the synaptic links is particularly challenging for recurrent neural networks (RNN), encoding input features and observed system dynamics within a continuous latent space. In this work, we investigate the representation built within the RNN while learning behavioral models of a class of discrete dynamical systems. To this end, we propose a method to extract the symbolic knowledge structured by the continuous state, based on Gaussian Mixture Model clustering. Experiments are performed on a pilot remanufacturing plant, by learning the model of a conveyor controller from process data. We show the capability of the RNN to achieve accurate predictions while providing a Moore Machine representation of the latent activations, consistent with the target system.**

***Index Terms*—Recurrent neural networks, Discrete systems, Model explainability, Industrial Cyber Physical Systems**

I. Context and motivation

Leveraging on the pervasive connectivity of smart devices and production processes, data-driven identification techniques are being considered more and more as a complementary approach to first principle modeling of Industrial Cyber Physical Systems (ICPS) [1], [2]. System identification algorithms shape models by fitting parameters using available observations, inherently support adaptation by including run-time data, and foster the digitalization of existing production systems lacking design-phase models [3]. In this context, modern deep neural networks are being widely investigated, due to their capability of structuring features by efficient representations [4].

The aim is to model the underlying data generator. However, the information extracted by the network is typically implicit and distributed across the neural connections, thus resulting almost impossible to be understood by the users. Indeed, the increase of explainability (i.e., achieve deeper insights into the mechanisms of the implemented network) is broadly recognized as a critical open issues within the deep learning research field [5]. Explainability is particularly challenging when employing Recurrent Neural Networks (RNN), including a stateful encoding of input features and observed system dynamics through hidden states, iteratively processed by implicit transition rules [6]. Continuous representation networks are employed to obtain differentiable architectures, so to exploit efficient learning algorithms. Nonetheless, in many practical applications, RNNs are intrinsically processing discrete features, from computer science (e.g., language, image captioning) to the manufacturing field (e.g., process mining, logical control). In these contexts, RNNs perform satisfactory inferences structuring a temporal symbolic knowledge in their continuous state space during training [6].

In the system identification field, RNNs encapsulate continuous representations of finite state machines when learning behavioral models of discrete time-state systems processing finite input-output signals, representing a broad class of practical applications. The extrapolation of such mechanisms could significantly support explainability.

Surprisingly, this topic dates back to 60s when Minsky demonstrated the relationships between RNN and Finite State Automata (FSA) in accepting regular languages [7]. Starting from this seminal study, theoretical models of computation have been exploited to investigate the capabilities provided by alternative prototype RNN architectures, mostly in terms of symbolic knowledge structuring by performing grammatical inference [6]. The major research momentum in this field occurred in 90s, starting from the observations that the orbits of internal activations tend to cluster when the network learns to emulate a FSA [8]. Several methods have been developed to extract FSA from RNN trained to perform grammatical inference, characterizing a class of algorithms often referred to as Rule Extraction (RE). A review on RE can be found in [9]. Despite having a long history, to the best of our knowledge, RE techniques have not yet been explored within the manufacturing field to achieve deeper insights into the representation learned by RNN-based ICPS models, as a way to increase explainability. In fact, most developments target grammatical inference, considering FSA processing sets of finite traces of symbols to assess their compliance with a target language, determined by single boolean output variable [10]. On the other hand, the learning of ICPS models requires to deal with concatenated multi-input multi-output data sequences generated by the continuous execution of the target process. Then, the hidden active state of the system has to be inferred from the temporized observations.

Leveraging on the theoretical framework provided by the

language inference field, in this work we propose a method to extract a human interpretable representation from RNNs trained to model the behavior of a class of discrete-time discrete-state dynamical system. To this end, we developed a rule extraction technique based on Gaussian Mixture Model clustering (GMM). Compared to alternative approaches proposed in the literature, GMMs provide a probabilistic interpretation supporting soft-assignments and the capability to tune cluster-specific full covariance matrices.

The proposed approach is applied to the conveyor of a pilot remanufacturing plant, for the identification of the behavioral model of the control system component of the ICPS [2]. We show that the trained RNN is able to infer the active state of the controller and predict the next action performed, encapsulating the finite state machine by its continuous latent space. We then extract the hidden Moore Machine (MM) representation by processing latent activations records. The extracted MM is exported toward a run-time environment to execute simulation scenarios on the same data processed by the continuous RNN, then the related input-output sequences are compared to analyze representation consistency.

The paper is organized as follows. Section 2 starts formalizing the focused system identification problem and linking to RNNs. Section 2A introduces the developed RE algorithm. Section 2B details the developed latent space clustering technique, based on GMM. Section 2C describes the adopted network architecture. Section 3 describes the targeted application and summarizes the results achieved.

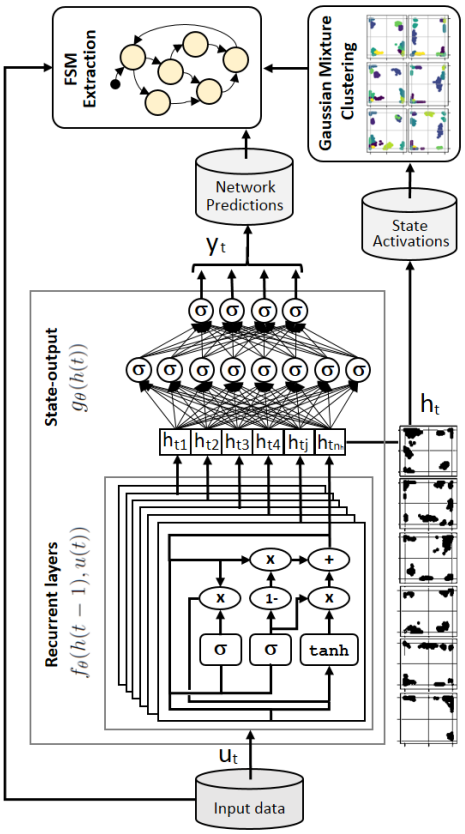

Fig. 1. Finite state machine extraction by clustering.

II. Method

In this work, we consider non-linear dynamic systems defined in discrete space and discrete time $t \in \mathbb{Z}$, driven by external signals $u(t)$, of the form:

$$\begin{cases} x(t) = p(x(t-1), u(t)) \\ y(t) = q(x(t)) \end{cases} \tag{1}$$

where $x(t) \in \mathcal{X}^{n_x} \subset \mathbb{Z}_+^{n_x}$ constitute the system state, being a finite subset of positive integer with $x(0) = s_0$, $u(t) \in \mathbb{B}^{n_u}$ the input set size, $y(t) \in \mathbb{B}^{n_y}$ the the output set size, with $\mathbb{B} = \{0, 1\}$. The nonlinear state-transition mapping is defined in $p[\cdot] : \mathcal{X}^{n_x} \times \mathbb{B}^{n_u} \rightarrow \mathcal{X}^{n_x}$, while the output mapping is defined in $q[\cdot] : \mathcal{X}^{n_x} \rightarrow \mathbb{B}^{n_y}$. n_x, n_u and n_y represents the size of states, input and output vectors size respectively.

From a system identification perspective, our aim is to infer a functional model of the unknown dynamic system through learning, by leveraging on the available input and output data sequences. Several techniques have been proposed in the literature for such purpose, including feedforward and recurrent networks based approaches. (see e.g., [11]). In this work, we exploit Recurrent Network architectures, which are receiving a lot of attention for system identification applications. The major strengths of RNNs reside in their capability to structure past input information within the hidden states, through loop connections. Besides, a compressed representation can be achieved by constraining the encapsulation of information from arbitrary long input sequence within a smaller size hidden state.

Formally, an RNN is a discrete time nonlinear dynamical systems defined as:

$$\begin{cases} h(t) = f_\theta(h(t-1), u(t)) \\ y(t) = g_\theta(h(t)) \end{cases} \tag{2}$$

where $h(t) \in \mathbb{R}^{n_h}$ characterize the RNN latent state size, which might result different from the dimension of the unknown state space of the target dynamical system. $f_\theta[.] : \mathbb{R}^{n_h+n_u} \rightarrow \mathbb{R}^{n_h}$, $g_\theta[.] : \mathbb{R}^{n_h} \rightarrow \mathbb{R}^{n_y}$ are parameterized nonlinear functions tuned through learning to fit the focused dynamical system behavior, minimizing the error between the network predictions and the given output targets.

A. Rule extraction method

As introduced above, RNNs are expected to encapsulate a continuous representation of the identified discrete system within its latent space. Thus, RNNs can be considered as a kind of neural state machine [6]. To extract the discrete representation embedded by the trained RNN, we propose the rule extraction algorithm summarized in Figure 1.

The algorithm proceeds as follows. The continuous state RNN is trained by backpropagation through time using the training set. Then, the network is executed in prediction mode,

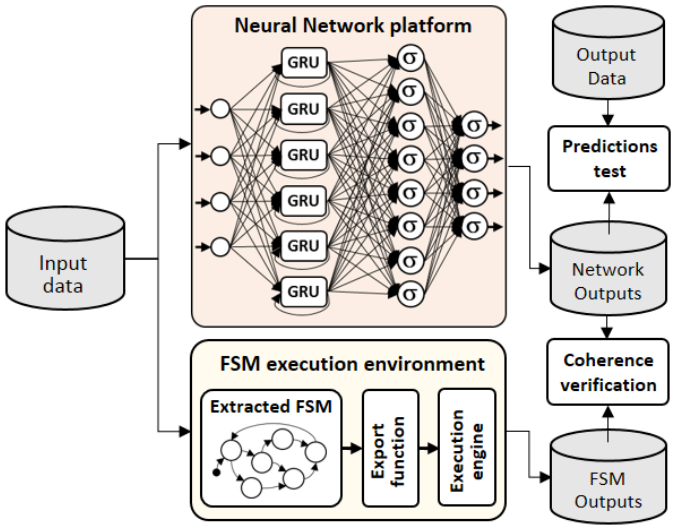

Fig. 2. Comparison of RNN and extracted FSM traces.

collecting both the outputs $y(t)$ and the latent state activations $h(t)$ related to each sample. Afterwards, the hidden activations records are processes by a clustering algorithm to identify discrete states within the continuous space. Indeed, as shown in [12], RNNs must structure their state space in subsets to achieve robust finite state computation (i.e. including injection of bounded additive noise to states). Moreover, the construction of a finite state representation in form of clusters is related to the capability of finding a proper set of stable attractors during training through bifurcation [13].
While performing rule extraction, each cluster is referred to the states of the induced state machine and performed trajectories, consequent to input sequence processing, to state-transition mappings. Afterwards, a state transition diagram is extracted, to visually represent the behavior of the RNN. Considering the characteristics of the problem at hand, we employ a Moore Machine (MM) diagram, formally defined as a 6-tuple $\{U, X, Y, \delta, \lambda, x_0\}$ where U, X, Y characterize the finite input, state and output sets and $\delta : U \times X \rightarrow X$, $\lambda : X \rightarrow Y$ state transition and output functions [14].

To evaluate the extracted MM, in this work we employ a simulation approach. The exploration of further techniques (e.g. by including model checking methods [15]) is foreseen for future extensions. The MM is compared with the RNN by running both on the same dataset and analyzing the output traces, as shown in Figure 2. To such an aim, the extracted MM is exported towards a simulation environment supporting executions: in this work, we employ Mathworks-Stateflow, representing a broadly used tool for state machines modeling. Eventual differences provide indications of inconsistencies. Then, further simulations can be performed, e.g. by changing the tuning parameters of the clustering algorithm, which is detailed described in the next section.

B. Latent space clustering technique

Several methods have been developed to extract symbolic knowledge in the form of finite state machine from RNNs. In [12], the state space is equally quantized in hypercubes followed by a breadth-first exploration of partitions. Notably, by adopting hyperbolic tangent activation functions, the points tend to concentrate on the corners of the hypercube $[-1, 1]^{n_h}$ if the hidden size results well balanced with the number of discrete latent states to be recover. Nevertheless, such assumption is not valid in general [9]. Vector quantization techniques have been proposed to tackle the exponential grows of partitions with the increasing latent space size, while providing clustering regions more fitted to the activations specifically occurring in the RNN. Several quantization algorithms have been investigated, including K-means (see e.g., [16]), Hierarchical Clustering (see e.g., [17]) and Self Organizing Maps (see e.g., [18]). In this work, we have developed a RNN latent space clustering algorithm based on Gaussian Mixture models. The major strengths resides in the full-covariance matrix support and soft-assignments, enabling the identification of partially overlapping non-isotropic clusters. Formally, the Mixture of Gaussian model of the latent activations is defined over a set of K multivariate Gaussian distributions $\mathcal{N}(x|\mu_k, \Sigma_k)$ and discrete latent variables z_k, representing the probability of an activation sample to belong to each component, as follows:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{3}$$

where $\pi \equiv \{\pi_1, \ldots, \pi_K\}$ represents the mixing coefficient and $\mu \equiv \{\mu_1, \ldots, \mu_k\}$ and $\Sigma \equiv \{\Sigma_1, \ldots, \Sigma_K\}$ the mean and covariance tensors.

The marginal density of the distribution of the activations then results:

$$p(x) = \sum_{k=1}^{K} p(z_k) p(x|z_k) \tag{4}$$

constituted by a factorized composition of component-wise conditional probabilities $p(x|z_k = 1) = \mathcal{N}(x|\mu_k, \Sigma_k)$ time the prior probability to sample each k-th component $p(z_k = 1) = \pi_k$. Then, by applying the Bayes rule, we obtain the posterior probability $p(z_k = 1|x) = \gamma(z_k)$, often referred to as responsibility, as:

$$\begin{aligned} \gamma(z_k) \equiv p(z_k = 1|x) &= \frac{p(z_k = 1) p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1) p(x|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)} \end{aligned} \tag{5}$$

which provides the soft assignment of each latent space activation point to the clusters characterizing the discrete states of the MM. To fit the parameters of the distribution, we iteratively employ the Expectation Maximization algorithm applying until convergence E-steps and M-steps:

$$\text{E-step}: \begin{cases} \gamma(z_{nk}) & = \dfrac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \end{cases}$$

$$\text{M-step}: \begin{cases} \mu'_k & = \dfrac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \\ \Sigma'_k & = \dfrac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu'_k)(x_n - \mu'_k)^T \\ \pi'_k & = \dfrac{N_k}{N} \\ N_k & = \sum_{n=1}^{N} \gamma(z_{nk}) \end{cases} \tag{6}$$

To select the number of components, we develop an integrated approach including Information criteria (i.e., Bayesian information criterion), cross validation, and silhouette indicators so to gain complementary insights regarding the learned RNN state discretization. Standard state minimization techniques might be considered to investigate the potential existence of equivalent minimal state machines [14].

C. Developed network architecture

Several RNN cells have been proposed to address the vanishing gradient issue of traditional units. Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) represent the most used in practical applications nowadays. Compared to LSTMs, GRUs employ a single gating unit to control the state update and the forgetting factor [19]. It has been shown that the former provides enhanced representation power, operating as an automata with external memory, at the cost of higher complexity in terms of parameters, whereas the latter behaves more as a finite state machine [20]. In this work, we implement an RNN based on GRU cells since well fitted with the characteristics of the problem at hand while computationally cheaper than LSTM. The output layer is defined as an element-wise sigmoid instead of the conventional softmax since the network must be capable to deal with multiple output actions independently activated at a certain time, thus representing a kind of multi-label classification problem.

Considering element-wise Bernoulli distributions over the output vector, we propose a binary-cross entropy objective function across the target actions to be predicted by the network, formally expressed as:

$$L = -\frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{n_y} [y_{i,j} \log(y_{i,j}) + (1 - y_{i,j}) log(1 - y_{i,j})] \tag{7}$$

with mini-batch size B and target actions y_t.

III. Results and Discussion

The proposed approach has been tested with the application to the STIIMA-CNR pilot plant for remanufacturing of mechatronic products [21]. We focus on the flexible conveyor system, aimed to move the of the Printed Circuit Boards (PCBs) across the operating stations (e.g. test, rework, etc.). Figure 3 reports a picture of the system. The conveyor is composed of 15 modular units with dedicated controllers, enabling agile layout reconfiguration following production needs.

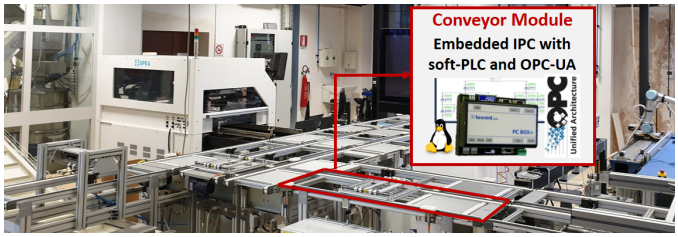

Fig. 3. Conveyor module of the remanufacturing plant.

The controllers are connected by an industrial Ethernet network and include an ISaGRAF Soft-PLC and an OPC-UA server, providing run-time data access and registration in databases. Modules execute equivalent instances of the control logic, supporting management of transfer tasks requests in each direction within the module. Figure 4 and Figure 5 display the Sequential Functional Chart (SFC) based control logic and the I/O list respectively.

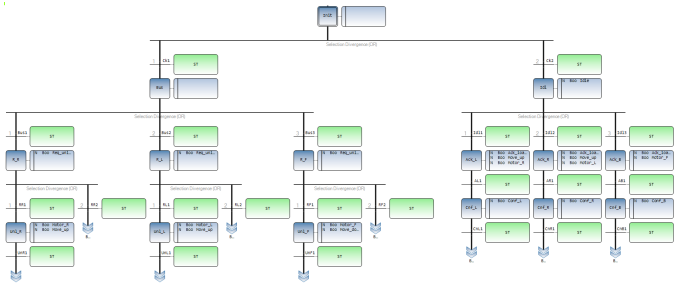

Fig. 4. SFC chart of the flexible conveyor system.

The dataset is constituted by a sequence of 10000 observations-actions pairs recorded from the PLC of a module , encoded as arrays of binaries. We implemented the neural network in Tensorflow 2.0. The RNN is trained in supervised learning to predict the next output signal of the module controller by observing a sequence of past input and output signals. For this reason, the dataset has been pre-processed to extract samples of I/Os records up to a certain time $t-1$ as the input set and the output at time t as the label. Then, the obtained dataset has been split into sub-parts including 60%, 30%, 10% of the samples, dedicated to training, validation and test respectively. The hyperparameter set include the number of layers of the network, the number of units in each layer, the extension backpropagation through time, the size of the mini-batch, the number of epochs and the patience of early stop routine. Then, a grid-search cross-validation approach has been adopted, employing the Adam algorithm for training the network by minimizing the objective function (8). A network architecture constituted by a GRU layer of 10 cells followed by a fully connected output layer has been adopted to process

Input signals

Name	Type	Description
Req_load_L	BOOL	Request to load from left
Ack_unload_L	BOOL	Acknowledge load from left
Received_L	BOOL	Received load from left
Idle_L	BOOL	Idle state to load from left
Req_load_R	BOOL	Request to load from right
Ack_unload_R	BOOL	Acknowledge load from right
Received_R	BOOL	Received load from right
Idle_R	BOOL	Idle state to load from right
Req_load_B	BOOL	Request to load from back
Ack_unload_F	BOOL	Acknowledge load from forw
Received_F	BOOL	Received load from forw
Idle_F	BOOL	Idle state to load from forw
Target_Left	BOOL	Pallet target = left
Target_Right	BOOL	Pallet target = right
Target_Forw	BOOL	Pallet target = forward
Reset_target	BOOL	Flag to reset pallet target
PalletPresent	BOOL	Pallet present in the module

Outputs signals

Name	Type	Description
Idle	BOOL	Module in idle state
Req_unload_R	BOOL	Request to unload right
Ack_load_R	BOOL	Acknowledgement to load right
Conf_R	BOOL	Confirm load right
Req_unload_L	BOOL	Request to unload left
Ack_load_L	BOOL	Acknowledgement to load left
Conf_L	BOOL	Confirm load left
Req_unload_F	BOOL	Request to unload forward
Ack_load_B	BOOL	Acknowledgement to load back
Conf_B	BOOL	Confirm load back
Motor_R	BOOL	Activate Motor move to right
Motor_L	BOOL	Activate Motor move to left
Motor_F	BOOL	Activate Motor move to forw
Move_up	BOOL	Move conveyor up for crossing
Move_down	BOOL	Move conveyor down to unlock

Fig. 5. I/O signals of the module controller.

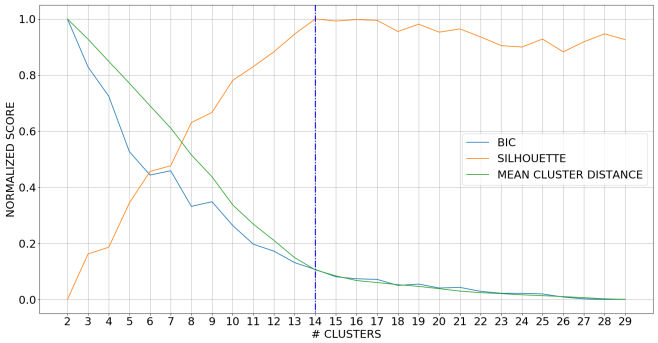

Fig. 6. BIC, Silhouette and Mean Cluster Distance curves

the test set. The further hyperparameters values and prediction results obtained are reported in Table 1.

Notably, the network reached 100% prediction accuracy, which is foreseeable when tackling deterministic problems by properly tuned learning machines and datasets covering the operating conditions of the target system. Nevertheless, the data driven model identified by the network is still a black-box. Therefore, we execute the trained network to collect the latent state activations observations while processing each sample. Then, the activations set has been processed by a GMM-based clustering procedure, implemented by means of Scikit-learn. Plots of the indicators to select the cluster size are reported in Figure 6, suggesting a choice of 14 components. The resulting clusters are reported in Figure 7, showing the cross activation of latent space neurons. Visibly, activation clusters are partially overlapping and non-isotropic.

Despite providing deeper view of the structure existing within the activation points inside the multidimensional continuous latent space, the cluster plot still result difficult to be interpreted by the user. Discrete states are clearly visible, but their relationship is still difficult to be understood. Moreover, this complexity increase with the size of the latent space. Figure 8 provides a more interpretable insight of the finite state representation embedded by the network through the continuous latent space. In details, the figure reports the Moore Machine obtained by processing aggregated input/clustered-state/output data and recording observed input conditioned state-transition and state-outputs within dedicated tensors. The Stateflow API is employed to construct the state machine programmatically. Visibly, the inferred state machine is equivalent to the SFC implementing the behavior of the conveyor module. It is worth noting that the self-transitions connected to some states of the Moore Machine are representing persistence in a state of the SFC when transitions conditions are not active. Despite validating the feasibility of the proposed method, such opportunity is not available in general. Indeed, comparing the extracted state machine with the one executed in the controller is typically not possible in practice, especially when performing data-driven ICPS models identification on brownfield plants. Therefore, we also checked the consistency of the extracted representation by comparing the state profiles of the RNN and the simulated state machine while processing the same data sequences. Figure 9 reports

TABLE I
HYPERPARAMETERS VALUES AND TEST SETS RESULTS

BPTT	*Mini-batch*	*Epochs*	*Patience*	*Pred. Accuracy*
50	64	50	5	100%

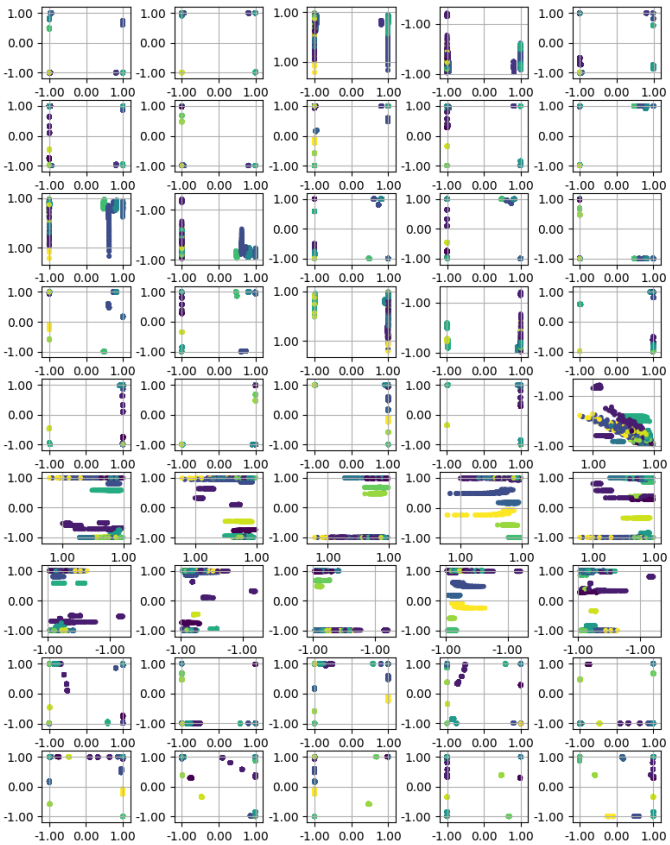

Fig. 7. Clustered latent space activity

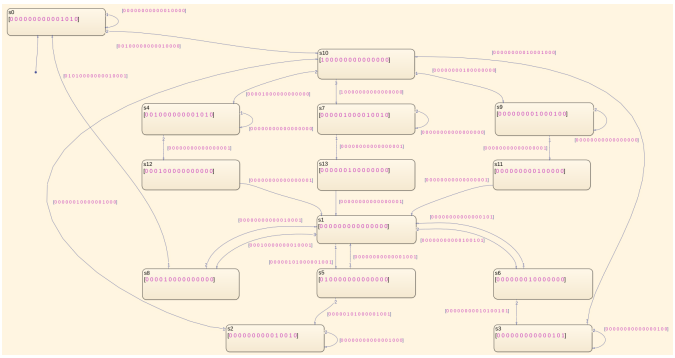

Fig. 8. Moore Machine constructed in Stateflow

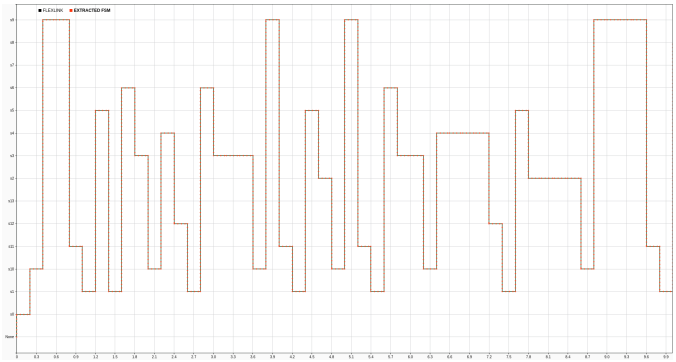

Fig. 9. MM machine (blue line) and RNN state prediction (dotted orange) sequences

an extract of the obtained state sequences, showing equal behavior. Such operation can be performed iteratively, e.g. by varying hyperparameters, while checking eventual changes in the acquired representation, providing a complementary form of analysis to conventional accuracy test. In fact, we found consistent representations while executing different runs of the experiments. This analysis represent a first step that we plan to extend in future works, e.g., by exploiting model checking techniques.

IV. Conclusion and Next Steps

In this paper we have proposed a method to extract the symbolic knowledge structured in the continuous state space of neural ICPS models. To such an aim, we processed the latent state activations of a trained Recurrent Neural Network with Gated Recurrent Units by a Gaussian Mixture Model based clustering technique. We showed the capability of the method to cluster the latent space activations, characterized by partially overlapping and non-isotropic distributions. Afterwards, a state transition diagram is extracted, providing a human interpretable representation of the symbolic knowledge structured by the RNN across the continuous latent space. Experiments have been performed by application to the conveyor of a pilot remanufacturing plant, learning the behavioral model of the control system component of the ICPS.

Our major aim is to achieve more explainable neural ICPS models, fostering the application of data-driven identification techniques in industrial context requiring deeper insights into the model to enable implementation. Such techniques are particularly beneficial when targeting brownfield plants where design phase models are not available.

Next developments will include the investigation of further class of discrete dynamical systems and hybrid systems, the integration of the clustering mechanism within the RNN layers to construct discrete latent states and the investigation of methods to validate the extracted representation.

References

[1] Y. Yuan, X. Tang, W. Zhou, W. Pan, X. Li, H.-T. Zhang, H. Ding, and J. Goncalves, "Data driven discovery of cyber physical systems," *Nature Communications*, vol. 10, 12 2019.

[2] P. Hehenberger, B. Vogel-Heuser, D. Bradley, B. Eynard, T. Tomiyama, and S. Achiche, "Design, modelling, simulation and integration of cyber physical systems: Methods and applications," *Computers in Industry*, vol. 82, pp. 273 – 289, 2016.

[3] L. Zhang, L. Zhou, L. Ren, and Y. Laili, "Modeling and simulation in intelligent manufacturing," *Computers in Industry*, vol. 112, 2019.

[4] D. Masti and A. Bemporad, "Representation of finite state automata in recurrent radial basis function networks," *2018 IEEE Conference on Decision and Control (CDC)*, 2018.

[5] High-Level Expert Group on AI, "Ethics guidelines for trustworthy ai," report, European Commission, Brussels, Apr. 2019.

[6] S. C. Kremer, *Field Guide to Dynamical Recurrent Networks*. Wiley-IEEE Press, 1st ed., 2001.

[7] M. Minsky, *Computation: Finite and Infinite Machines*. Prentice-H, 67.

[8] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland, "Finite state automata and simple recurrent networks," *Neural Comput.*, vol. 1, pp. 372–381, Sept. 1989.

[9] H. Jacobsson, "Rule extraction from recurrent neural networks: A taxonomy and review," *Neural Comput.*, vol. 17, June 2005.

[10] Q. Wang, K. Zhang, A. G. Ororbia II, X. Xing, X. Liu, and C. L. Giles, "An empirical evaluation of rule extraction from recurrent neural networks," *Neural Computation*, vol. 30, no. 9, pp. 2568–2591, 2018.

[11] A. Brusaferri, M. Matteucci, P. Portolani, and S. Spinelli, "Nonlinear system identification using a recurrent network in a bayesian framework," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, pp. 319–324, July 2019.

[12] C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun, and Y. C. Lee, "Learning and extracting finite state automata with second-order recurrent neural networks," *Neural Computation*, vol. 4, no. 3, 1992.

[13] P. Tino, W. Horne, and C. Giles, "Fixed points in two–neuron discrete time recurrent networks: Stability and bifurcation considerations," 2001.

[14] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[15] M. Mazzolini, A. Brusaferri, and E. Carpanzano, "Model-checking based verification approach for advanced industrial automation solutions," in *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pp. 1–8, Sep. 2010.

[16] S. Das and M. Mozer, "Dynamic on-line clustering and state extraction: An approach to symbolic learning," *Neural Networks*, vol. 11, no. 1, pp. 53 – 64, 1998.

[17] A. Sanfeliu and R. Alquezar, "Active grammatical inference: A new learning methodology," in *in Shape, Structure and Pattern Recogniton, World Scientific Pub*, pp. 191–200, 1994.

[18] P. Tino and J. Sajda, "Learning and extracting initial mealy automata with a modular neural network model," *Neural Computation*, 1995.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT, 2016.

[20] C. Wang and M. Niepert, "State-regularized recurrent neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 6596–6606, PMLR, 09–15 Jun 2019.

[21] A. Brusaferri, E. Leo, L. Nicolosi, D. Ramin, and S. Spinelli, "Integrated automation system with pso based scheduling for pcb remanufacturing plants," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, pp. 990–995, July 2019.