Extracting State Machines from Feedforward Neural Networks

Sebastian Seidel
Bundeswehr University Munich -
Computer Science Department
Neubiberg, Germany 85577
sebastian.seidel@unibw.de

Uwe M. Borghoff
Bundeswehr University Munich -
Computer Science Department
Neubiberg, Germany 85577
uwe.borghoff@unibw.de

***Abstract*—Artificial neural networks (ANNs) are a category of AI models based on the principles of connectionism that have recently been successfully used to solve problems that were not solvable with classic AI approaches in the past. The aim of this paper is to introduce a method to extract state machines from Feedforward Neural Networks (FNN) at a specific point of training to provide an approach in Inter-cognitive communication for better understanding the logical decision processes that are simulated by a FNN's calculations. This method is directly fitted to be used with FNNs that calculate decisions over continuous input data, where the future situation is based on the recent situation and the decisions that are made in the present. Based on the extraction from FNN's, themselves being a basis for many other neural network types, our method is supposed to deepen the link between connectionist's models and symbolic models, thus improving the usability of artificial neural networks data processing in the environment of symbolic human concepts.**

I. Introduction

Artificial neural networks have seen a renaissance in the past few years, when they were used to solve a variety of problems that cannot be formally described well and are therefore not well suited to be solved by classic AI approaches [4]. This became possible because of the improved hardware available nowadays and the vast amount of data available for training. But one major problem still remains. It is often very hard to verify or even understand the decision processes on which the results computed by artificial neural networks are based. We want to address that problem by providing equivalent state machines for FNNs which operate over continuous input data. Those state machines can be analyzed instead of those neural networks, thus providing a Representation-sharing transfer type for Inter-cognitive information transfer between FNNs as artificial cognitive systems and humans as natural cognitive systems according to the definition in [1]. We choose FNNs, because they are the basic structure for a big variety of commonly used artificial neural networks like Recurrent Neural Networks or Convolutional Neural Networks [6]. We also restrict our method to continuous input data, because with this type of input a future situation is based on its predecessor. This is comparable to state machines, were a future state depends on its predecessor in contrast to, e.g. decision trees. With this approach we want to provide a link between commonly used connectionists' and symbolic AI models by creating a transformational hybrid system as they are described in [3], that transforms FNNs into equivalent state machines. Our aim is to improve human understanding of decision processes in artificial Neural Networks, particularly when they show an exceptional behavior, by providing a CogInfoCom Interface for developers designing and training FNNs [2]. Such an interface can afterwards be used by Domain Experts that evaluate ANNs to enhance approaches for ANN training like the one described in [11]. This approach is visualized in Figure 1.

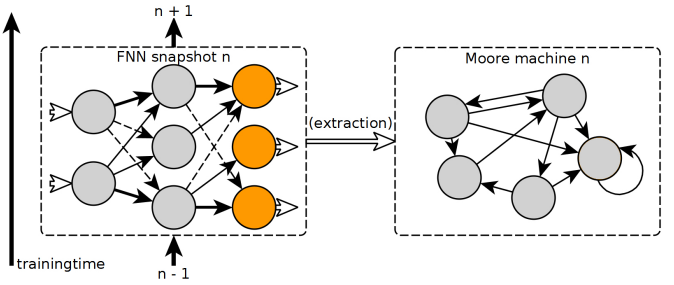

Fig. 1. The basic idea of extracting state machines from FNN snapshots in training time, when the FNN shows an exceptional good or bad decision behavior.

The paper is structured in five sections. In Section II, an overview of related work is given. In Section III, we describe the problems related with the extraction of state machines from FNNs. In Section IV, we introduce our approach and discuss limitations. Section V concludes our paper.

II. Related Work

In this section we will introduce the filler/role principle of Smolensky's Integrated Connectionist/Symbolic Cognitive Architecture that serves as inspiration for our method. We will further provide a short description of our approach to extract symbolic characteristics from FNNs, something we need to define our states and transitions.

A. The filler/role principle

According to Smolensky it is possible to establish a connection between connectionists AI and symbolic AI [8]. He states with the filler/role principle that the complete meaning of a symbol like a letter in a word is defined by its filler, a picture or placeholder, and its role or, in other words, its context. In the case of a letter in a word this is its position in that word compared to those of all other letters. Both can be combined in a distributed representation [7], [9].
We also use the idea of distributed representations for filler and context to identify states in FNNs.

978-1-5386-7094-1/18/$31.00 ©2018 IEEE

B. Extracting characteristics from Feedforward Neural Networks

The second basis for the method described in this paper is our approach to extract decision trees from FNNs. The key element to accomplish this extractions is the extraction of characteristics that are combined to receive the single categories forming a FNNs output. As a result, for a given FNN with given output categories, one can determine which characteristics have to be or must not be present to receive a specific classification category as output. Further, one can determine all input activation vectors that cause the presence of a specific characteristic. The overall result is that for each possible output all valid input activation vectors that lead to that output can be determined [12]. We use this result to define the states of our extracted state machines. This idea is based on the approach to to define classes for ANN inputs [10].

III. Statement of the Problem

In this section, we will point out the primary challenges we have to cope with when extracting state machines from a FNN and the questions that result from these challenges.
In contrast to the decision trees we extracted from FNNs [12], state machines do not only receive information from the outside world but additionally have access to information about their internal state. Therefore, we have to extract the following information from a FNN to create an equivalent state machine:

- The states a system can have and how these states are characterized.
- The transitions between those states consisting of their starting state, their target state, and the signal or command that causes the transition.

By answering these two questions we can generate the complete graph describing the state machine. To achieve that goal we make use of the input signals of the FNN and their corresponding output activation vectors. With those inputs and corresponding outputs we define the states of the state machine. After defining all possible states we search for all directed connections and their triggering signals which, when combined, define the transitions between the states. Finally, we have to review whether all generated transitions are usable or whether some of them can never be triggered by valid input activation vectors. When this is done we propose that the extracted state machine simulates the FNN's behavior under the condition that this network is only fed with valid input signal vectors.

IV. The Proposed Method

In this section, our method to extract a state machine from a FNN will be presented. First we describe how to generate the states. Second we evaluate what the characteristic activation combinations belonging to the specific states are. Third we generate all possible transitions that can potentially connect the states of the state machine. Fourth all transition that can never be triggered by valid input data are eliminated. The result is a state machine, which shows a behaviour equivalent the the behaviour of the original FNN.
When extracting state machines we always extract Moore machines, because in Moore machines the output symbols are only dependent on the state they are in. This way, we can treat a Moore machine's state as directly connected to the output signals. In other words, if a system reaches a new state, this state is defined by the produced output. We demonstrate our technique to extract a Moore machine from a FNN using an example agent for a small task and specify the single steps in the process.
For this example we assume an agent being implemented with the help of a small FNN. Its neurons can be either active or inactive. The agent should be able to walk along a corridor with several doors. It cannot turn, just walk straight forward, and recognizes only the door closest to it. The doors can be closed or open, and if they are closed, they can additionally be locked. An open door is never locked. The FNN has two output neurons, enabling the agent to choose from four possible actions with the following activation combinations:

relevant output-neuron	activation 'unlock door'	activation 'open door'	activation 'close door'	activation 'step forward'
neuron 'X'	0	0	1	1
neuron 'Y'	0	1	0	1

The agent is supposed to walk along the corridor. When it finds a door in front of it the agent can walk trough, if the door is open. If the door is closed, it first has to open this door. If the door is locked, the agent has to unlock the door befor it can open the door. When the agent passed a door and the door, now behind him, is still open, the agent has to close the door before continuing to walk along the corridor. The basis of the agent's decisions is the information the FNN receives with its three input-neurons. These neurons have the following activation combinations:

encoded information	neuron 'A'	neuron 'B'	neuron 'C'
'door in front of agent'	1	*	*
'door behind agent'	0	*	*
'door is open'	*	1	1
'door is closed'	*	0	*
'door is locked'	*	0	0
'door is unlocked'	*	*	1

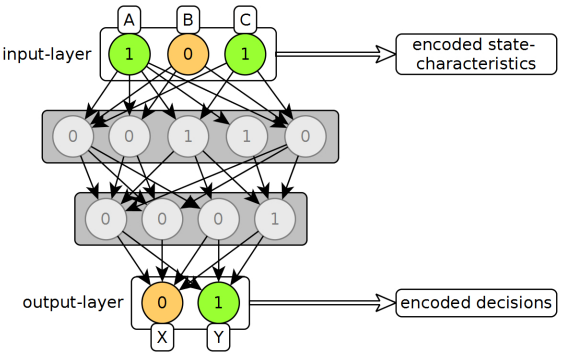

Fig. 2. An example for a FNN that fulfills all requirements to realize the example agent from Section IV.

The equivalent FNN might look like the example pictured in Figure 2. We assume this represents the neural network at a

promising point of its training were it computes correct results significantly often.

A. Generating the states

Now we can start with the extraction of the corresponding Moore machine by extracting its states from the FNN, which realizes the function f_{FNN} by deriving specific output activation vectors from specific input activation vectors. We generate the states according to the following description:

Step 1) We consider $O_k = (o_1, ..., o_n)$ as a possible output vector with $o_1, ..., o_n$ being the activations of the n ordered neurons in the output layer, representing the order in $(o_1, ..., o_n)$. Determine the set of all corresponding input vectors $I_k = \{I_s = (i_1, ..., i_m) \mid f_{FNN}(I_s) = O_k\}$ with $i_1, ..., i_m$ being the activations of the n ordered neurons in the output layer, representing the order in $(i_1, ..., i_m)$. This is done exactly the same way as it was done for the characteristics we extracted for our decision trees [12]. The definition for the resulting states is $state_k = I_k$. They are representing the categorized combinations of activations in the output-layer.

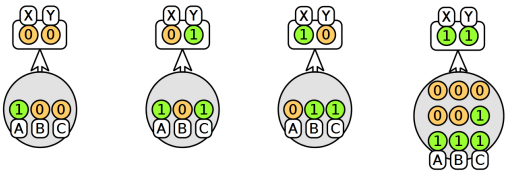

Fig. 3. The assumed state-output-combinations for the FNN in the example from Section IV.

The categorized activation combinations are of interest to us because they are encoding the output of the Moore machine we want to extract. Remember that the output symbol in a Moore machine only depends on the state the Moore machine is in. Therefore, we have the same number of states as we have different output symbols. The output symbols themselves are represented by the mentioned activation combinations of the output-layer. To be precise, each specific state is directly connected to exactly one specific output symbol. The second important trait of states is that each single one is defined by a set or combination of characteristics that can change over time [5]. That leads us to the statement that the following traits are true for states:

- Each specific state is connected to exactly one specific output symbol defined by a definite pattern of activations in the output layer.
- Each state of a system is connected to a defined number of specific sets or combinations of characteristics that system can have.

As a result each valid pattern of activations in the output layer presenting a state is directly connected to a number of specific sets or combinations of characteristics the system can have. That is why we are searching for the matching combinations described in step 1). Let us assume that our analysis provides the following results for our example FNN. Those results lead to the combinations of states with outputs presented in Figure 3. It is important to note that the system can actually be in any of those states because the artificial neural network has a valid output combination matched to each one of these input activation vectors building that states.

neuron 'A'	neuron 'B'	neuron 'C'		neuron 'X'	neuron 'Y'
1	0	0	⇒	0	0
1	0	1	⇒	0	1
0	1	1	⇒	1	0
0	0	0	⇒	1	1
0	0	1	⇒	1	1
1	1	1	⇒	1	1

B. Finding the characteristic activation combinations of each state

In our next step, we try to reduce the information overhead in the input activation vectors that form the states of our future Moore machine. Therefore, we are searching for input neuron activation combinations that are typical for a specific state. These activation combinations cannot be found in any input activation vector that does not exclusively belong to that state. As a direct result, an input activation vector can be identified as part of a specific state by just using these input-neuron activations because they can only be found in this state. All other activations of such an activation vector are irrelevant for the assignment of the activation vector to the state and can be treated as random valid activations. We will always try to reduce activation vectors to activation combinations as small as possible and therefore searching for a minimum of joint neuron activations to identify all sets of relevant neuron activations for a state. With this method, we also identify all combinations of characteristics that define this state by eliminating information and characteristics that are not considered for that state.
In our example, we can use that method to optimize the characterization of the fourth state presented in Figure 3. The activation vector $(1, 1, 1)$ cannot be reduced. The only activation it shares with any of the other activation vectors is the third, which is the same activation as in $(0, 0, 1)$. But the last active neuron having the activation 1 is also true for activation vectors in the second and third state shown in Figure 3. Therefore the activation combination $(1, 1, 1)$ is one complete valid combination for the fourth state. On the other hand, the activation vectors $(0, 0, 0)$ and $(0, 0, 1)$ can be reduced. They share the joint activation combination $(0, 0)$ for their first and second neuron. This activation combination for those two neurons is not present in the activation vector of any other state. When the first two neurons of an activation vector have the activation value '0' the activation value of the third neuron holds no additional information regarding the membership of this activation value to a specific state. This vector is always part of the fourth state, because that combination cannot occur in any other state and all possible activation combinations for the other neurons are valid combinations in the fourth state combined with the activation values $(0, 0)$ for the first two neurons. Consequently, the following activation value combinations are defining characteristics for the fourth state:

Remember our definition that a door, which is open, always has to be unlocked, too. The reduction described for our example is additionally visualized in of Figure 4. There you

activation combination	neuron 'A'	neuron 'B'	neuron 'C'
'door closed & behind'	0	0	*
'door open & front'	1	1	1

can see the complete reduction process for the fourth state presented in Figure 3. As a result, we frame our second step to extract a Moore machine:

Step 2) The value(i_x) is the activation value of a specific input activation i_x in $(i_1, ..., i_m)$, $state_k = \{l_s = (i_1, ..., i_m)$, $values_i = \{v_i\}$ is the set of all valid values for input activations and a, b $\in \{1, ... , m\}$. Reduce the required characteristics to completely describe a state by applying the following rules step by step to all sets $T = \{l_s\}$ with $|T| = |\{v_i\}|$:
If $\forall a\ (\forall v_i\ (\exists l_s\ (value(i_a) = v_i))) \wedge \forall b\ (\exists v_i\ (\forall l_s\ (value(i_b) = v_i)))$ then set for all i_a $(value(i_a) = '*')$. $\exists i_x \in l_s \in T \mid x \neq a \wedge x \neq b \rightarrow$ delete T. We define c_R = number of indices b in the vectors in set R. For all $R \in \{T\}$ with $c_R = \max\{c_T\}$ combine all $l_s \in R$ to one vector and for all $l_s \in R$ delete all other T with $\exists l_s \in T$.
Repeat this procedure until no sets T are left.

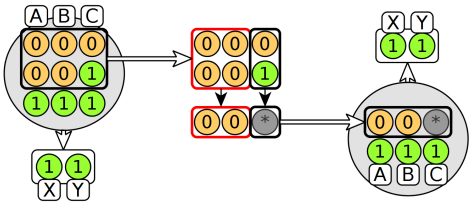

Fig. 4. Reducing the information overhead in activation vectors by identifying activation combinations that are characteristic for specific states.

C. Generating all possible transitions

The next step is to find at least one transition for each possible state, leading from that state to another possible state. To accomplish this goal we need to take a closer look at the differences between the specific states. Because states are defined by their sets of characteristics, which are represented by input activation vectors, the similarity of two states is encoded in those sets of vectors. For example in Figure 3, the difference between the first and the second state is just the one single activation value of input neuron 'C', which can be either active ('1') or inactive ('0'). Therefore it requires only the single input value from 'C' switching from inactive to active to change from the first to the second state. We will call them similiar with vectorial distance 1 [7]. In our method we are only looking for transitions from one state to its most similar state. The vectorial similarity of two states is bound to the similarity of that pair of activation vectors, one of each state, that have the closest vectorial distance. If there are two states with the same similarity to a third state we establish transitions from both of them.
We are looking for this distance because the input for a FNN operating over continous input data, if not manipulated, changes gradually if time steps small enough are used for the input update. The consequence is that an input activation vector will always switch to the vectorial closest input activation vector it reaches with its changed activations. As an experiment, imagine to reduce the time for input actualization until only one valid activation value per time step is changed for only one valid step of change. The input vector will only change one activation value step by step, until it reached the next vectorial closest activation vector of another state it can reach with that sequence of changes. At this point, the whole system is in the next state. This leads us to the third step for the extraction of Moore machines.

Step 3) $S = \{state \mid state = \{l_s = (i_1, ..., i_m)\}$ is the set of all states and $E = \{edge_{ij}\}$ is the set of edges between all $states_i \in S$ and all $states_j \in S$. Additionally, $C = \{connection_{hk} = (edge_{hc}, ... , edge_{dk}) \mid connection_{hk} = \min\{||state_h - state_c|| + ... + ||state_d - state_k||\}\}$ is the set of all direct or indirect connections with the shortest euclidian distance from $state_h$ to $state_k$. Create all transitions made possible by the activation vectors of the states, using the following algorithm:
$\forall(state_i, state_j \in S)\ \nexists(connection_{ij} \in C)$ calculate the shortest euclidian distance between the activation vectors in $state_i$ and $state_j$ with $\exists l_r \in state_i \wedge \exists l_t \in state_j \wedge (||state_i - state_j|| = ||l_r - l_t|| = \min\{|||l_u - l_v|| \mid u, v \in \{1, ... , m\}\})$. Add $edges_{pq}$ and $edges_{qp}$ with $||state_p - state_q|| = \min\{||state_i - state_j||\}$ as new transition to the Moore machine and update E and C. For $state_p$ the relevant vector is $l_r \in state_p = (i_{p_1}, ... , i_{p_m})$, for $state_q$ it is $i_t \in state_q = (i_{q_1}, ... , i_{q_m})$. Calculate the symbols for those transitions with $symbol_{pq} = \{i_{q_z} \mid i_{q_z} \neq i_{p_z}\}$ where i_{q_z} is a single input activation and $z \in \{1, ... , m\}$. Repeat this process until every state is at least connected to one other state.
Additionally, each state has internal transitions which lead to itself. They are at least needed if the FNN can receive the same input activation vector two times in a row.

For our example, the resulting transitions and symbols connecting the extracted states are pictured in Figure 5.

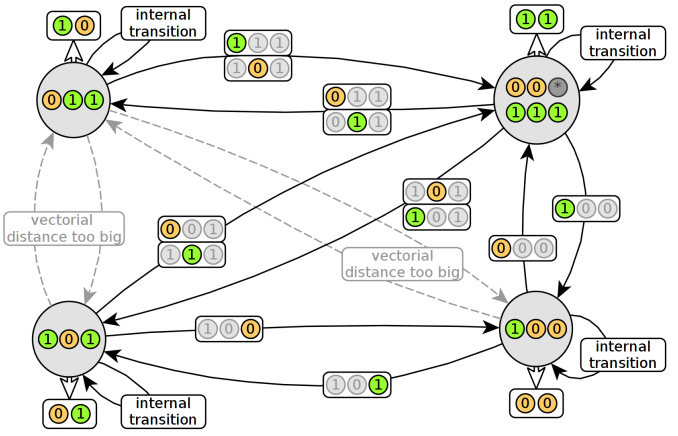

Fig. 5. The first iteration of possible transitions between the states and their triggering symbols, consisting of changing neuron activations.

D. Eliminating all transitions that are not based on valid input activation vectors

At this point, we covered all possible transitions the FNN implements to reach every other state from a random starting state with as few changes in the input vector as possible.

There can be transitions among them that are possible if only the vectorial distances are considered but cannot be triggered by a valid, non-manipulated sequence of input vectors. We consider it a valid successor input activation vector when a vector has only a minimum of changes in its input activations compared to its predecessor's input activations combined with the changes the FNN itself causes by its output activations. The reason for the requirement of minimum changes is the continuous input data we assume for our method. Therefore, the input describing a situation or system can only undergo small changes in a sufficiently short period of time. So, a valid successor input activation vector represents the characteristics its predecessor had combined with the changes the outside world and the FNN's output can cause in a small time period. To identify those transitions that should never be activated, we calculate those new input vectors that can be reached. They can be determined by applying the state's outputs to the input activation vectors that form the respective state. This forms the components that have to be part of all valid successor activation vectors. For example, in our extracted Moore machine the state defined by the vector $(1, 0, 1)$ creates the output $(0, 1)$, which causes the system to open the next door. Therefore, the next valid input activation vector has to incorporate 1 as second activation, because this is the encoding of the next door being open. That again forces all outgoing transitions from that state to incorporate a 1 as second activation, too, because in $(1, 0, 1)$ it is encoded that the door is closed, which has to be changed. As a result, all outgoing transitions not matching this requirement can be deleted. They will never be triggered by valid, not-manipulated input data. The complete elimination for our example is displayed in Figure 6. All activations changing in a transition from one state to another state are colored in this figure, while all unchanged activations are grey, defining the context in which those changes happen. The combination of both represents the symbol triggering the transition.

The following two aspects are important when dealing with more complex ANNs. First, the change in the input activations that an output causes can effect several activations over a number of timesteps, for example, if a door is opened in realtime, the input consists of a 1024x768 picture, and the input refresh rate is a millisecond. The decisive changes and resulting components for the outgoing transitions are those that appear, based on the actual output, in the next millisecond. Second, it is important that we view the output's effects as components we have to take into account when calculating the new activations and not pure activations themselves. In more complex ANNs a number of those components might occur at the same timestep and superimpose each other to form those activations the valid successor vectors have to incorporate.

If the input activation vectors are not directly encoding semantic values but pictures, sound waves, or other data, the calculation of the new activations that have to be incorporated by the successor vector can be done with the help of a simulation. With the help of a suitable interface the responsible programm receives the description of the recent states corresponding situations (input activation vectors) and the output, applies the output to those situations, and evaluates the resulting successor situations. These are then translated back into the valid successor activation vectors and with them the related valid transition symbols. A suitable simulation for a game's FNN would be the game itself or for machine control the corresponding training programm for the FNN.

The resulting fourth step to extract a Moore machine from an FNN, removing all transitions that can never be triggered, includes therefore the following tasks:

Step 4) Compute all valid successor input activation vectors $I_{s_{n+1}}$ in timestep n+1 for all input activation vectors I_{s_n} | $f_{FNN}(I_{s_n}) = O_{k_n}$ forming the recent state in timestep n. This is done by applying the function $f_{next}(I_{s_n}) : I_{s_n} \text{ x } O_{k_n} \rightarrow I_{s_{n+1}}$ to all I_{s_n} in the recent state.

Evaluate all valid transition symbols by evaluating the activations defining their changes and contexts. This is done by comparing the activations of the vectors I_{s_n} forming the recent state with the activations of those valid successor vector $I_{s_{n+1}}$, which has the shortest euclidian distance to the respective vector. $I_{s_{n+1}}$ must include all activation changes in the symbol and for all other activations those activations in I_{s_n}.

Delete all transitions whose symbols were not evaluated.

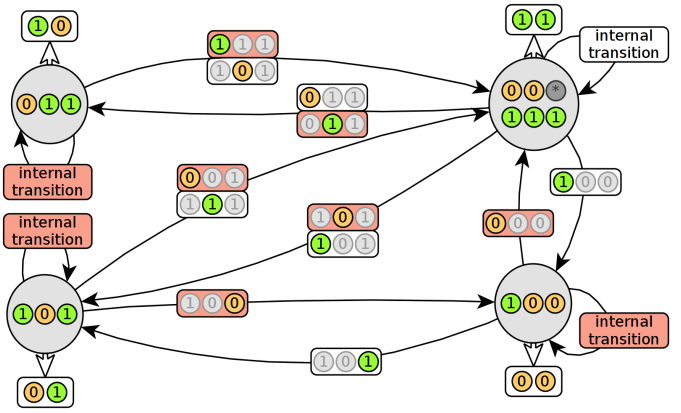

Fig. 6. The extracted Moore machine with all transitions that can never be triggered by valid inputs marked red.

After all transitions that cannot be triggered are removed, the extracted Moore machine matches the behavior simulated by the original FNN provided that it is only fed with valid, not manipulated input data.

One form of manipulation can be to cut a sequence of input activation vectors out of the input stream. That would be the equivalent of jumping from one state to another one. But we have to keep in mind that those jumps can occur if the FNN receives that type of manipulated data. This is because the network simulates the behaviors like the extracted Moore machine but it still works according to the mechanics of artificial neural networks.

Although the Moore machine is now completely extracted, the overall evaluation can still be difficult for bigger Moore machines with complex activation vectors. Therefore, the states and transition symbols should be tagged with appropriate names. This tagging has to done manually, because the ANN and Moore machine can assign activation vectors to input and output signals, but both have no information how those signal combinations (symbols) are named in a human language. The states can be simply named after the actions their respective output causes. They are formed by a series of input data connected to their activation vectors, e.g., pictures that all trigger this output. E.g., in our Moore machine $(0, 1)$ is 'open

door' or $(1, 1)$ is 'step forward'. The transitions can be named after the situational change its changing activations correspond to, completed by the contextual information its unchanging activations encode, e.g., the part where two pictures differ. In our Moore machine $(0, 1, 1)$ is named 'looking away from door'. The completely extracted and tagged Moore machine for our example is depicted in Figure 7.

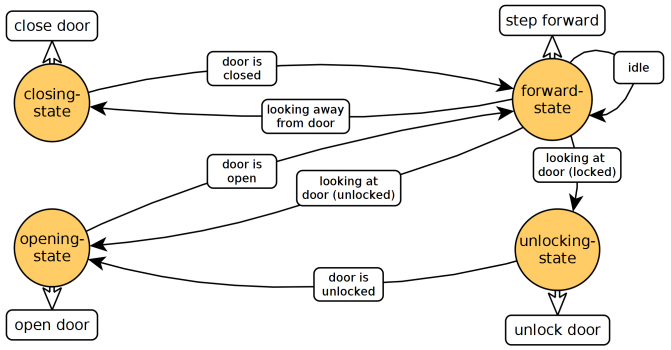

Fig. 7. The final Moore machine extracted from the example artificial neural network pictured in Figure 2.

E. Shortcomings of the proposed method

Our approach to extract Moore machines from FNNs uses the same principle to receive all possible input activation vectors for given classifications and their output vectors as the method to extract decision trees from FNNs that we presented [12], which, at the moment, restricts us to FNNs. Therefore, besides being restricted to the chosen models, our approach has the following limitations:

restriction	considered improvements	possible related problems
only 'active' & 'inactive' are valid activation values	incorporating real-valued activations & still search for thresholds	computational overhead might become extremely large
arithmetical & logical operations required	incorporating tensor-based filler/role multiplication	limitations regarding the neural network models that can be processed

To overcome these restrictions and largely avoid the related possible problems is still left for future work.

V. Conclusion

In this paper we described a method to extract Moore machines from FNNs, based on the idea of calculating corresponding input activation vectors for given characteristics in the output and consequently for given output vectors. The resulting Moore machine works according to symbolic rules but simulates the FNNs behavior under completly valid input data. Considering the fact that Moore machines are one of the most common symbolic AI and automation models and that FNNs are the basis for many types of artificial neural networks, this approach offers a useful option for a better understanding of specific artificial neural networks continuous-input-data-based decision processes with the help of symbolic analysis. Especially when a FFN shows an exceptional good or bad overall performance or always fails to solve only one specific situation correctly, it might be usefull to extract the corresponding Moore machine and evaluate the succesfully trained behavior or a missbehavior with established symbolic analysis methods.
The given shortcomings may have a significant impact on the practicability of our proposed method, but they can be solved despite these solutions may come at the cost of increased computational overhead. To avoid that increase in computational overhead will be one of the major challanges when improving our approach in the future, when we will implement our method in a small tool to create a usable CogInfoCom Interface [2] and test it with a set of FNNs.
Nevertheless, we think our approach provides a good enhancement for the idea of extracting symbolic models from artificial neural networks as a method to connect artificial neural networks and classic, symbol-based AI models.

Acknowledgment

The authors would like to thank Wolfgang Hommel, Michael Grabatin, Michael Steinke and Franz Schmalhofer for polishing our paper, helping us to identify potential drawbacks and their open-minded discussions regarding our ideas. We highly appreciate their support.

References

[1] Péter Baranyi, Ádám Csapó, *Definition and Synergies of Cognitive Infocommunications* Hungary: Obuda University - Acta Polytechnica Hungarica, vol. 9, no.1, pp.67-83, 2012.

[2] Péter Baranyi, Ádám Csapó, *CogInfoCom Channels and Related Definitions Revised* IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, pp.73-78, 2012.

[3] Kenneth McGarry, Stefan Wermter, John MacIntyre, *Hybrid Neural Systems: From Simple Coupling to Fully Integrated Neural Networks* Neural Computing Surveys, vol. 2, pp.62-93, 1999.

[4] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning* Cambridge, Massachusetts: The MIT Press, 2017.

[5] Stuart Russel, Peter Norvig, *Artificial Intelligence: A Modern Approach - Third Edition* Upper Saddle River, New Jersey: Pearson Education, Inc., 2010.

[6] Fjodor van Veen, *The Neural Network Zoo* Utrecht, Netherlands: The Asimov Institute, 2016. https://www.asimovinstitute.org/neural-network-zoo/

[7] Paul Smolensky, Géraldine Legendre, *The harmonic Mind - Volume 1: cognitive architecture* Cambridge, Massachusetts: The MIT Press, 2011.

[8] Paul Smolensky, *Connectionist AI, Symbolic AI, and the Brain - Artificial Intelligence Review* Heidelberg, Germany: Springer-Verlag GmbH, 1987.

[9] Paul Smolensky, *Symbolic functions from neural computation* London, England: The Royal Society Publishing, 2012.

[10] Karen Simonyan, Andrea Vedaldi, Andrew Zisserman *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps* Cornell University Library - arXiv:1312.6034, 2014.

[11] Aline Dobrovsky, Uwe M. Borghoff, Marko Hofmann, *Applying and Augmenting Deep Reinforcement Learning in Serious Games through Interaction* Periodica Polytechnica Electrical Engineering and Computer Science, v.61, no.2, pp.198-208, 2017.

[12] Sebastian Seidel, Sonja Schimmler, Uwe M. Borghoff, *Understanding neural network decisions by creating equivalent symbolic AI models* to be published at: London, England: Intelligent Systems Conference, 2018.