

Explaining Deep Learning Models Through Rule-Based Approximation and Visualization

Eduardo Soares, Plamen Angelov, *Fellow, IEEE*, Bruno Costa, *Senior Member, IEEE*, Marcos P Gerardo Castro, Subramanya Nagesh Rao, Dimitar Filev, *Fellow, IEEE*

Abstract—This paper describes a novel approach to the problem of developing explainable machine learning models. We consider a Deep Reinforcement Learning (DRL) model representing a highway path planning policy for autonomous highway driving [1]. The model constitutes a mapping from the continuous multidimensional state space characterizing vehicle positions and velocities to a discrete set of actions in longitudinal and lateral direction. It is obtained by applying a customized version of the Double Deep Q-Network (DDQN) learning algorithm [2]. The main idea is to approximate the DRL model with a set of *IF...THEN* rules that provide an alternative interpretable model, which is further enhanced by visualizing the rules. This concept is rationalized by the universal approximation properties of the rule-based models with fuzzy predicates. The proposed approach includes a learning engine composed of 0-order fuzzy rules, which generalize locally around the prototypes by using multivariate function models. The adjacent (in the data space) prototypes, which correspond to the same action are further grouped and merged into so-called "MegaClouds" reducing significantly the number of fuzzy rules. The input selection method is based on ranking the density of the individual inputs. Experimental results show that the specific DRL agent can be interpreted by approximating with families of rules of different granularity. The method is computationally efficient and can be potentially extended to addressing the explainability of the broader set of fully connected deep neural network models.

Index Terms—Deep Reinforcement Learning, explainable AI, rule-based models, prototype- and density-based models, density-based input selection, autonomous driving.

I. INTRODUCTION

RESEARCH on self-driving vehicles have made significant progress in recent years. However, a challenging topic in the field of self-driving cars concerns the transparency of the trained machine learning models which is needed for validation, verification, and certification [3]–[8]. The demand of understandable models involves interpretability and explainability of a trained agent in order to fully understand the knowledge encoded in them.

Linguistic *IF ... THEN* fuzzy rule-based models offer transparent insights in contrast to neural networks rather 'black box' approaches [9], [10]. Although these 'black box'

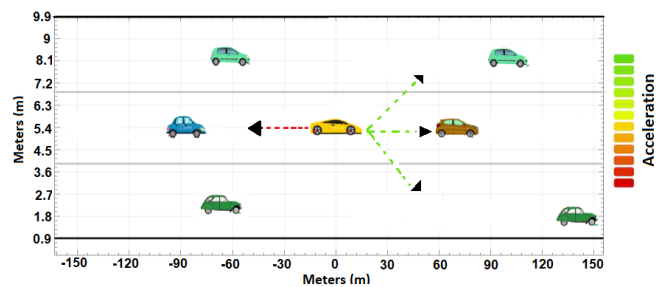


Fig. 1. Example of host (*ego*) and surrounding vehicles on a highway, where the host vehicle is represented by the center vehicle (yellow car). The forwards arrows indicate the possible directions which the *ego* vehicle can move. The backwards arrow indicates the brake maneuver.

models reach impressive classification and approximation accuracy, their nested non-linear structure makes them highly non-transparent [11], [12].

This paper proposes a new explainable self-organizing approach to transform a trained deep neural network model into a set of *IF ... THEN* rules. We use a Deep Reinforcement Learning (DRL) model of the path planning policy for highway self-driving [1] to simulate data corresponding to driving scenarios. The model maps the set of continuous state variables characterizing the position and velocities of the *ego* vehicle (EV) and the surrounding vehicles on a divided highway into a set of discrete actions in longitudinal and lateral direction.

State variables include meaningful affordance indicators of the road situation such as the longitudinal and lateral position and velocity of the host vehicle and relative longitudinal and lateral positions and velocities of the surrounding vehicles. The output of the model is a set of eight possible decisions/actions in longitudinal (maintain, accelerate, brake, and hard brake) and lateral (lane keep, change lane to right, and change lane to left) directions - Fig. (1).

The main idea of this paper is to provide an approximation of the DRL model with an alternative interpretable model with a similar performance. Our approach is based on the following main concepts: i) the universal approximation ability of the rule-based models with fuzzy predicates; ii) the better interpretability of the prototype-based fuzzy rules (including visualization). The method allows to potentially learn the rules incrementally during the DRL training process.

The proposed method for learning rules expands from the previously published work [13], [14] by introducing a

* Corresponding author

E. Soares* and P. Angelov are with LIRA, School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK, e.almeidasoares@lancaster.ac.uk, p.angelov@lancaster.ac.uk.

M. Castro, S. Nagesh Rao and D. Filev are with Ford Research and Innovation Center, Ford Motor Company, Palo Alto, CA 94304, USA, dfilev@ford.com, mgerard8@ford.com, snageshr@ford.com.

B. Costa, No Affiliation, brunuxcosta@gmail.com.

density-based method for selecting the most important inputs and a two-stage hierarchical approach to group the adjacent prototypes in the data space that correspond to the same action. These two novel techniques allow us to reduce the number of prototypes needed and improve the explainability. This is achieved both linguistically as a set of hierarchical *IF...THEN* rules and through visualisation. In addition, we also propose a sequence of pair-wise decision process, rather than one decision for the recommended action, and a method for balancing the training data set to have approximately the same data samples per action. In combination, these innovations allow us to get an explainable approximation of the DRL agent decisions under multiple driving conditions and to summarize its performance in diverse situations.

In order to validate our concept, experiments were conducted using the DRL model provided by Ford Motor Company, see [1] for details. Results demonstrate that the proposed approach can achieve a computationally efficient, compact and easily explainable approximation of DRL models.

The focus of the paper is on the methodology for modeling of the multidimensional data set that is obtained through approximation of the simulated DRL model. The method is not constrained to DRL models and can be extended and adapted to other type of deep learning structures and architectures as well as to learning from data generated by human drivers.

The remainder of this paper is structured as follows. Section II introduces the proposed method. The data employed in the analysis is presented in Section III. The results and the discussion are provided in Section IV. Section V concludes the paper.

II. THE PROPOSED APPROACH

In this section, we will introduce the approach briefly describing the general architecture, learning and validation. Let $T = \{(\mathbf{x}_k, \mathbf{a}_k)\}_{k=1}^N$ be training data set with $\mathbf{x}_k \in \mathbb{R}^n$ denoting the state vector and $\mathbf{a}_k \in \{1, \dots, A\}$ denoting the action vector for each $k \in \{1, \dots, N\}$. The layered architecture (Fig. (2)) can be seen as a mapping, $f : \mathbb{R}^n \rightarrow \mathbb{R}^A$; n is the number of inputs; A is the number of actions; i is the specific data sample/point k ; N is the number of training data samples. Separate learning cycles are introduced for each action. Therefore, the data set is split into A sub-sets.

The learning process starts with analyzing the mutual proximity of the data [15]. As a result, a small number of prototypes are being selected which are actual data samples that are most representative locally. When prototypes are being formed only data samples that correspond to the same action are being considered. When prototypes that correspond to different actions are being put together in the data space a further level of analysis is being made, namely merging adjacent (in the data space) prototypes that correspond to the same action together forming so-called "*MegaClouds*". Finally, the *MegaClouds* can be visualized and also represented by *IF...THEN* rules. The general architecture of the proposed approach is given in Fig. (2).

As a result, we compose A parallel *IF...THEN* rules, each of which corresponds to one of the A actions and has the following form:

$$R_l : IF (x \sim p_l^1) OR (x \sim p_l^2) OR \dots OR (x \sim p_l^{P_l}) \quad (1) \\ THEN (action \ l)$$

where p_l^j ($j \in \{1, \dots, P_l\}$) is the j^{th} prototype of the l^{th} action; P_l is the number of identified prototypes that represent the l^{th} action.

The identified prototypes are connected with logical "OR" (implemented as a t -conorm). Strictly speaking, each of the conditions within the *IF...THEN* rules are fuzzy rules on their own but all of them have the same consequent pointing to the same action. The so-called "*winner takes all*" principle is used to decide the action of the *IF...THEN* rule during the validation process.

In summary, the proposed method can be represented as a hierarchy (see Fig. (3)) where the bottom layer is the data set and the next layer up is composed of all the identified prototypes during the learning process, while the top layer of the structure consists of a much smaller sub-set of highly informative prototypes corresponding to *MegaClouds*.

In the following two subsections, the main steps of learning and validation are described.

A. Learning rules from the data

The proposed method learns the prototypes associated with each action in a separate loop. Therefore, the data set is split during the training into sub-sets.

As each *IF...THEN* rule is identified separately for each action, unless specifically declared otherwise, all the mathematical notations in the algorithm consider the l^{th} action by default and the index l is omitted for clarity.

Step 1. Standardize the newly observed data sample, \mathbf{x} . Standardization is performed on a per input basis:

$$\hat{x}(i) = \frac{x(i) - \mu(x(i))}{\sigma(x(i))} \quad (2)$$

where $\hat{x}(i)$ denotes the standardized value of the i -th input of the data sample; $\mu(x(i))$ denotes the mean of the i -th input and $\sigma(x(i))$ denotes the standard deviation of the i -th input; $\boldsymbol{\mu} \in \mathbb{R}^n$ denotes the vector of the mean values and $\boldsymbol{\sigma} \in \mathbb{R}^n$ denotes the vector of the standard deviations.

Following the standardization the data is being normalized converting it to the range $[0, 1]$. Unity-based normalization of the i -th input is given by [16]:

$$\bar{x}(i) = \frac{\hat{x}(i) - \min(\hat{x}(i))}{\max(\hat{x}(i)) - \min(\hat{x}(i))} \quad (3)$$

where $\bar{x}(i)$ denotes the normalized values of the i -th input.

Step 2. Initialize the algorithm meta-parameters with the first data sample, \bar{x}_1 observed:

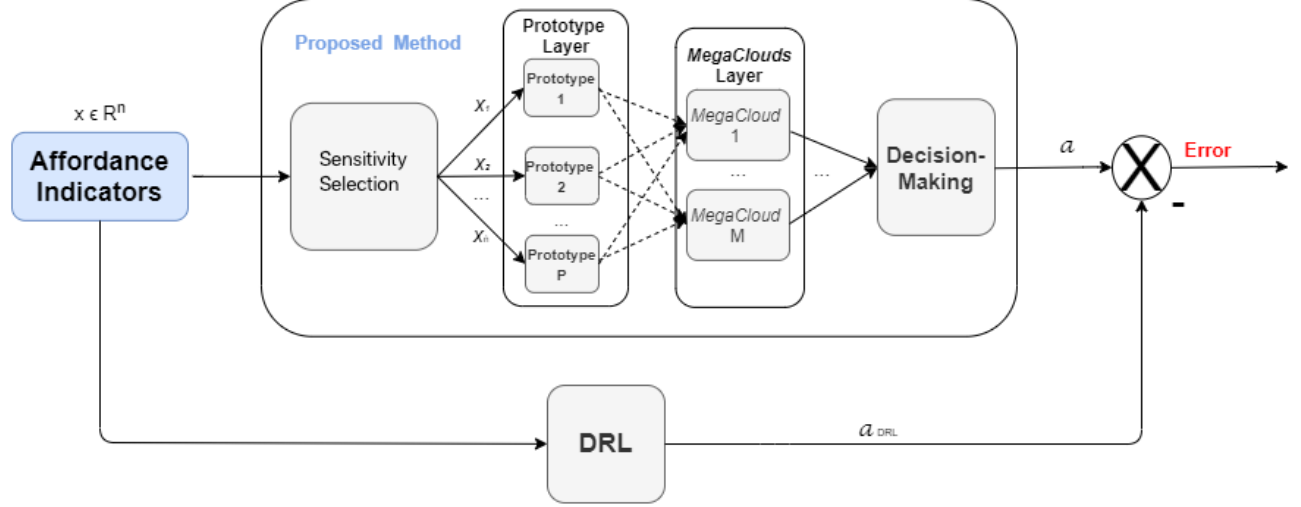


Fig. 2. General structure of the proposed approach. a_{DRL} refers to the DRL output. The comparison between a and a_{DRL} is used to determine the accuracy of the proposed method.

$$\begin{aligned} \mu_1 &\leftarrow \bar{x}_1; & P &\leftarrow 1; & p^1 &\leftarrow \bar{x}_1 \\ C^1 &\leftarrow \{\bar{x}_1\}; & S^1 &\leftarrow 1; & r^1 &\leftarrow r^o; \end{aligned} \quad (4)$$

These include: i) the mean being initialized with the first normalized data point; ii) the number of prototypes being set to 1 ; iii) the first prototype being initialized with the first data point; iv) initialize the first so-called *data cloud*, C^1 as a set of data points that are associated with the first prototype (*data clouds* were defined in [17] as set of data points described by a prototype and differ somewhat from clusters by shape and boundaries and other properties); v) the so-called *support* of the *data cloud* S^1 defined as the number of data points associated with a certain *data cloud* [15]; vi) the radius of the area of influence around the prototype, r^1 , in this paper we initialize it with $r^o = 0.5$. In a multidimensional space $r^o = 0.5$ is reasonable (not too low to avoid getting significant number of prototypes and not too high to allow a certain level of detail and granularity). Notice that r^1 is the only meta-parameter, its value is automatically determined by the algorithm. However, it is not user-defined and problem-specific as it only refers to an initial value which later will be updated with the real data.

Based on this initialization define the first *IF...THEN* rule for the given (l -th action) as follows:

$$\mathbf{R}_l : \text{IF } (x \sim p_l^1) \text{ THEN } (\text{action } l) \quad (5)$$

Step 3. Calculate the data density at the current data point, \bar{x}_k ; $k \in \{1, \dots, N\}$. Starting from the mutual distances (Euclidean or Mahalanobis type) between the data points (samples) in the feature space it can be demonstrated theoretically [9] that the data density takes the form of a Cauchy type function as in Eq. (6).

$$D(\bar{x}_k) = \frac{1}{1 + \frac{\|\bar{x}_k - \mu_N\|^2}{\|\sigma_N\|^2}}; \quad (6)$$

where D is the data density, and σ_N denotes the standard deviation.

In this step we also identify the prototype p^{j^*} that is nearest to \bar{x}_k :

$$j^* = \underset{j \in \{1, \dots, P\}}{\operatorname{argmin}} \{ \|\bar{x}_k - p^j\|^2 \} \quad (7)$$

Then, using the density and the distance to the nearest prototype, p^{j^*} , we check the following condition [9] based on which we determine if the current data point is going to be added to the set of prototypes or not. If the condition is met, go to **Step 4**; otherwise, go to **Step 5**.

$$\begin{aligned} \text{IF } (D(\bar{x}_k) \geq \max_{j \in \{1, \dots, P\}} (D(p^j))) \\ \text{OR } (D(\bar{x}_k) \leq \min_{j \in \{1, \dots, P\}} (D(p^j))) \\ \text{OR } (\|p^{j^*} - \bar{x}_k\| > r^{j^*}) \\ \text{THEN } (\text{add a new data cloud}) \end{aligned} \quad (8)$$

where $D(p^j)$ is density of the nearest prototype, p^j .

Step 4. Add a new data cloud:

$$\begin{aligned} P &\leftarrow P + 1; & C^P &\leftarrow \{\bar{x}_k\}; \\ p^P &\leftarrow \bar{x}_k; & S^P &\leftarrow 1; r^P &\leftarrow r^o; \end{aligned} \quad (9)$$

Then, go to **Step 6**.

Step 5. Update the meta-parameters of the nearest data cloud:

$$\begin{aligned} C^{j^*} &\leftarrow C^{j^*} + \{\bar{x}_k\}; \\ p^{j^*} &\leftarrow \frac{S^{j^*}}{S^{j^*} + 1} p^{j^*} + \frac{S^k}{S^{j^*} + 1} \bar{x}_k; \\ S^{j^*} &\leftarrow S^{j^*} + 1; \\ r^{j^*} &\leftarrow \sqrt{\frac{(r^{j^*})^2 + \|\sigma^{j^*}\|^2}{2}}; \end{aligned} \quad (10)$$

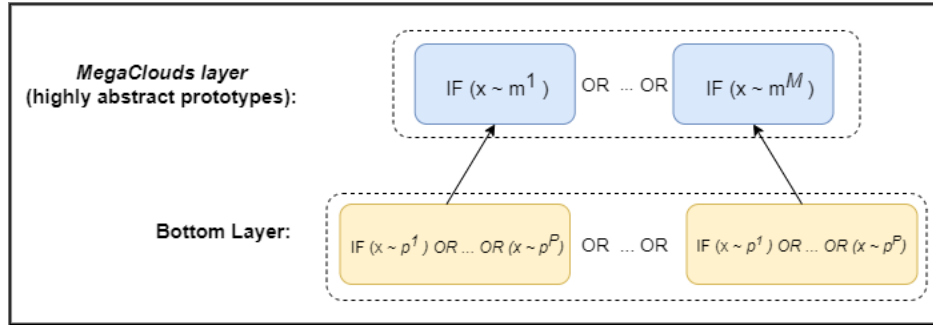


Fig. 3. Hierarchical structure - *MegaClouds*, where m^M is the mean of the M -th *MegaCloud* associated with the l^{th} action

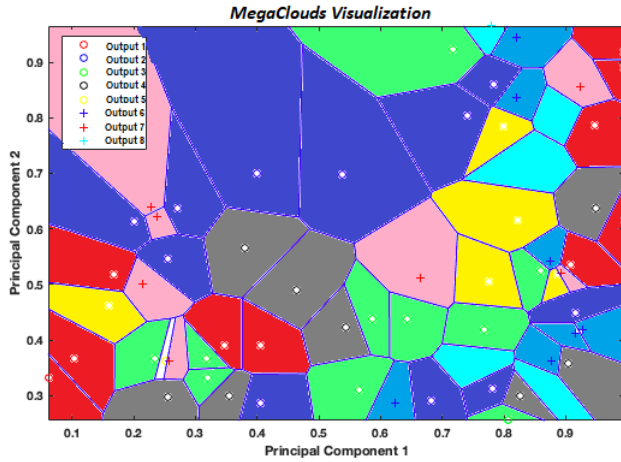


Fig. 4. *MegaClouds* visualization in terms of Voronoi Tesselation

- Action 6 (Lane change left and also brake by $-2m/s^2$): 213 samples
- Action 7 (Lane change right): 7704 samples
- Action 8 (Lane change right and also brake by $-2m/s^2$): 102 samples

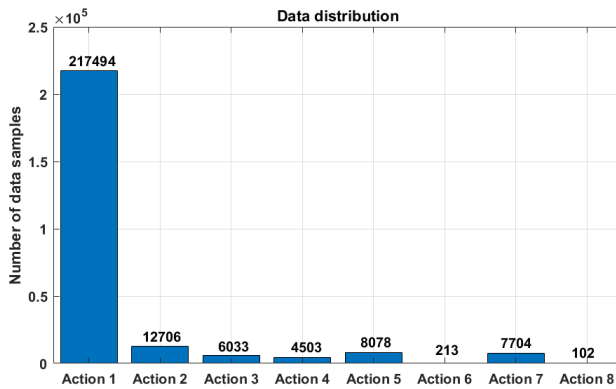


Fig. 5. Data Distribution in terms of different maneuvers/actions by the ego vehicle, showing the clearly data imbalance nature of the the data set.

The data set provided by Ford Motor Co. was obtained by a simulating DRL model representing driving policy of a

TABLE I
DESCRIPTION OF THE INPUTS

| Inputs | Description |
|--------|---|
| 1 | Ego lateral position |
| 2 | Relative velocity between ego and center vehicles |
| 3 | Front left vehicle position longitudinal |
| 4 | Front left vehicle velocity |
| 5 | Front left vehicle lateral position |
| 6 | Front center vehicle position longitudinal |
| 7 | Front center vehicle velocity |
| 8 | Front center vehicle lateral position |
| 9 | Front right vehicle position longitudinal |
| 10 | Front right vehicle velocity |
| 11 | Front right vehicle lateral position |
| 12 | Rear left vehicle position longitudinal |
| 13 | Rear left vehicle velocity |
| 14 | Rear left vehicle lateral position |
| 15 | Rear center vehicle position longitudinal |
| 16 | Rear center vehicle velocity |
| 17 | Rear center vehicle lateral position |
| 18 | Rear right vehicle position longitudinal |
| 19 | Rear right vehicle velocity |
| 20 | Rear right vehicle lateral position |

self-driving vehicle in diverse traffic conditions. More details can be found in [1].

The data set was divided into 80% for training and 20% for validation purposes as usual for such tasks [21]. We used 10-fold cross validation for the experimental setup. It is important to highlight that the analyzed dataset is imbalanced as illustrated in Fig. (5). However, due to the prototype-based nature of the hierarchical approach no pre-processing is required in this case.

A. Performance Evaluation

In order to evaluate the performance of the proposed method the accuracy index is considered. Accuracy is defined as follows:

$$ACC(\%) = \frac{TP + TN}{TP + FP + TN + FN}, \quad (16)$$

where TP, FP, TN, FN denote true and false, negative and positive respectively.

All the experiments were conducted with MATLAB 2018a using a personal computer with a 1.8 GHz Intel Core i5

TABLE II
PERFORMANCE COMPARISON FOR DIFFERENT ACTIONS AND NUMBER OF INPUTS

| # Inputs ¹ \ Accuracy | Action 1 | Action 2 | Action 3 | Action 4 | Action 5 | Action 6 | Action 7 | Action 8 | Overall |
|----------------------------------|--------------|--------------|--------------|--------------|---------------|--------------|--------------|---------------|---------------|
| 20 | 98.46% | 85.03% | 75.52% | 49.46% | 80.16% | 66.66% | 92.96% | 50.0% | 94.54% |
| 10 | 98.75% | 88.02% | 83.06% | 91.34% | 91.72% | 72.72% | 96.85% | 38.09% | 97.41% |
| 7 | 99.8% | 93.3% | 98.5% | 94.3% | 90.02% | 92.7% | 95.4% | 66.7% | 98.94% |
| 5 | 98.71% | 83.97% | 81.66% | 78.06% | 87.91% | 80.00% | 95.84% | 76.47% | 96.75% |
| 3 | 98.36% | 84.43% | 77.92% | 39.92% | 78.98% | 81.08% | 93.06% | 66.67% | 94.89% |

TABLE III
PERFORMANCE COMPARISON FOR DIFFERENT ACTIONS WITH 7 INPUTS WITH THE HIGHEST DENSITY

| Method \ Accuracy | Action 1 | Action 2 | Action 3 | Action 4 | Action 5 | Action 6 | Action 7 | Action 8 | Overall |
|----------------------------------|--------------|--------------|--------------|--------------|---------------|---------------|---------------|--------------|---------------|
| This paper | 99.8% | 93.3% | 98.5% | 94.3% | 90.02% | 92.7% | 95.4% | 66.7% | 98.94% |
| This paper (<i>MegaClouds</i>) | 99.34% | 89.82% | 82.9% | 81.45% | 92.76% | 72.0% | 97.31% | 72.72% | 97.88% |
| SVM | 88.65% | 53.6% | 0% | 0% | 100.0% | 100.0% | 74.7% | 0% | 87.08% |
| KNN | 97.34% | 83.1% | 85.6% | 84.7% | 72.6% | 100.0% | 90.2% | 70.00% | 95.23% |
| Decision Tree | 98.82% | 88.16% | 83.4% | 82.26% | 93.36% | 82.92% | 92.45% | 70.00% | 97.02% |
| Adaboost | 94.3% | 72.9% | 88.0% | 97.9% | 87.9% | 0% | 86.0% | 0% | 92.95% |
| Discriminant analysis | 91.0% | 42.3% | 33.9% | 0% | 36.6% | 32.8% | 38.6% | 10.3% | 85.43% |
| Random Forest | 99.4% | 90.2% | 88.6% | 98.1% | 94.2% | 75.2% | 86.0% | 75.4% | 98.31% |
| XGBoost | 99.2% | 84.5% | 86.9% | 87.3% | 89.9% | 32.8% | 89.4% | 0% | 97.12% |
| CatBoost | 93.2% | 78.7% | 90.1% | 92.2% | 88.2% | 52.3% | 87.1% | 0% | 91.45% |

TABLE IV
DESCRIPTION OF THE 7 INPUTS WITH HIGHER DENSITIES

| Inputs | Description |
|----------|---|
| Rv | Relative velocity between Ego and center vehicles |
| d^{FL} | Front left vehicle position longitudinal |
| V^{FL} | Front left vehicle velocity |
| d^{FC} | Front center vehicle position longitudinal |
| d^{FR} | Front right vehicle position longitudinal |
| V^{FR} | Front right vehicle velocity |
| d^{BC} | Rear center vehicle position longitudinal |

of the proposed approach. It allowed to infer a meaningful approximation of the DRL model and enabled quick evaluation of its performance for specific use cases. Table V details the number of prototypes produced by the proposed recommendation system considering different layers and 7 inputs (best scenario).

Generated trapezoidal fuzzy rules for the *MegaClouds* layer (highly abstract layer) can be illustrated in terms of inputs as illustrated by Fig. (9). It also can be visualized in terms of rules per prototype as given by Fig. (10).

Fig. (11) illustrates the actions given by the proposed method along the time. This is helpful to analyse the driving behavior and sequence of events by specialists.

In general, experiments have shown that the proposed explainable method is an efficient framework for this challenging task. Results showed advantages (98.94%) compared to similar methods for addressing the approximation task. Moreover, the proposed method in its top layer also produced transparent linguistic fuzzy rules, which are human

interpretable. In addition, the hierarchical architecture allows to reduce the rule antecedents and to simplify the structure of the rule-based models.

V. CONCLUSION

In this paper, we propose a novel explainable rule-based machine learning model that can be used to approximate the decisions policy of a DRL agent. To generate training data we used a DRL model representing a highway path planning policy for autonomous driving. The model is composed of a 0-order fuzzy rules. Experiments have shown that the proposed method was able to produce more accurate results than the other similar state-of-the-art methods.

We also present a new hierarchical mechanism to significantly reduce the number of generated fuzzy rules. In this case, adjacent (in the data space) prototypes which correspond to the same action are grouped and merged into so-called "*MegaClouds*". The proposed method helped to improve the interpretability of the generated models. Moreover, the input selection method based on ranking the density per input dimension in the data space contributed to improve the accuracy of the models as it creates individualized subsets of inputs per action, taking advantage of the parallel characteristic of the proposed explainable self-organizing method. Experimental results show that an accurate and computationally efficient explainable alternative to the deep neural network model can be successfully developed providing opportunities to explain and validate the decisions by the DRL agent.

¹Inputs with the highest density

TABLE V
NUMBER OF IDENTIFIED PROTOTYPES PER ACTION FOR DIFFERENT HIERARCHICAL LAYERS

| Layer \ # Prototypes | Action 1 | Action 2 | Action 3 | Action 4 | Action 5 | Action 6 | Action 7 | Action 8 | Total |
|----------------------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|
| Bottom Layer | 1315 | 1009 | 482 | 360 | 649 | 17 | 607 | 4 | 4443 |
| <i>MegaClouds</i> | 13 | 14 | 8 | 10 | 15 | 6 | 11 | 4 | 81 |

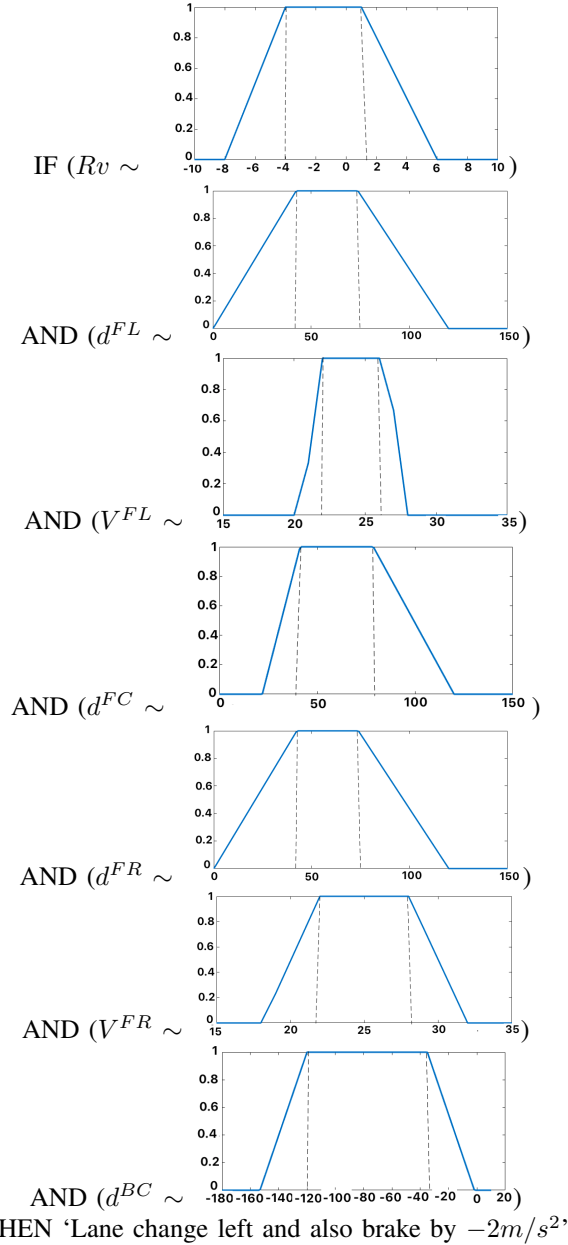


Fig. 9. Trapezoidal rule per feature for 'Lane change left and also brake by $-2m/s^2$ ' (Action 6)

ACKNOWLEDGMENT

The first two authors would like to thank Ford Motor Company for the financial and technical support. This work is part of the URP project "Transparent Deep Learning Classifier of Driving Scenarios able to Identify and Learn

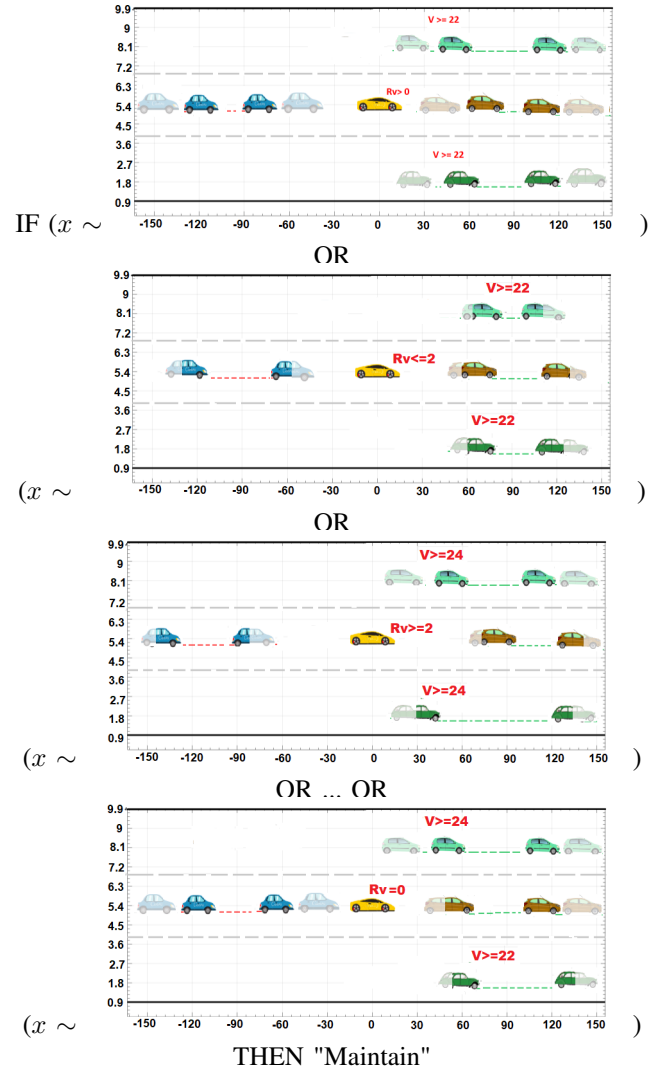


Fig. 10. Visual interpretation of trapezoidal rule for "Maintain" (Action 1), where the watermarked cars represent the soft trapezoidal fuzzy boundaries and the solid cars denotes the limits of the *MegaClouds*. R_v denotes the relative velocity between EV and center vehicle, and V denotes the velocities for the front left and front right vehicles, both R_v and V are in m/s

from Unseen Situations".

REFERENCES

- [1] S. Nagesh Rao, H. E. Tseng, and D. Filev, "Autonomous highway driving using deep reinforcement learning," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 2326–2331.
- [2] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

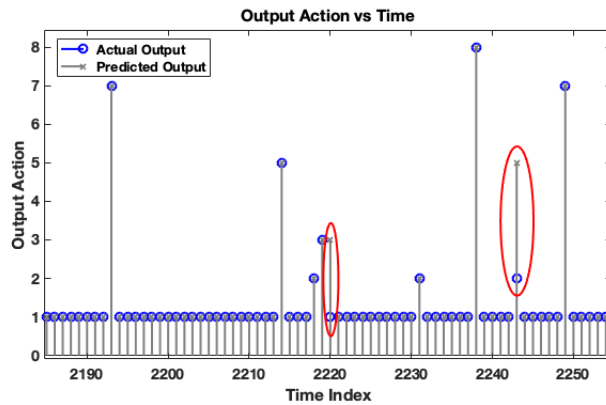


Fig. 11. Action vs time. The red ellipsoid indicates the wrongly predicted action given by the proposed approach

- [3] M. Coeckelbergh, "Artificial intelligence, responsibility attribution, and a relational justification of explainability," *Science and engineering ethics*, pp. 1–18, 2019.
- [4] G. Wiegand, M. Schmidmaier, T. Weber, Y. Liu, and H. Hussmann, "I drive-you trust: Explaining driving behavior of autonomous cars," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–6.
- [5] S. Ramos, S. Gehrig, P. Pinggera, U. Franke, and C. Rother, "Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 1025–1032.
- [6] J. Stilgoe, "Machine learning, social learning and the governance of self-driving cars," *Social studies of science*, vol. 48, no. 1, pp. 25–56, 2018.
- [7] I. Škrjanc, G. Andonovski, A. Ledezma, O. Sipele, J. A. Iglesias, and A. Sanchis, "Evolving cloud-based system for the recognition of drivers' actions," *Expert Systems with Applications*, vol. 99, pp. 231–238, 2018.
- [8] G. Gutierrez, J. A. Iglesias, F. J. Ordoñez, A. Ledezma, and A. Sanchis, "Agent-based framework for advanced driver assistance systems in urban environments," in *17th International Conference on Information Fusion (FUSION)*. IEEE, 2014, pp. 1–8.
- [9] P. P. Angelov and X. Gu, *Empirical approach to machine learning*. Springer, 2018, ISBN: 978-3-030-02384-3.
- [10] J. A. Iglesias, P. Angelov, A. Ledezma, and A. Sanchis, "Evolving classification of agents' behaviors: a general approach," *Evolving Systems*, vol. 1, no. 3, pp. 161–171, 2010.
- [11] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.
- [12] G. Andonovski, G. Mušič, S. Blažič, and I. Škrjanc, "Evolving model identification for process monitoring and prediction of non-linear systems," *Engineering applications of artificial intelligence*, vol. 68, pp. 214–221, 2018.
- [13] P. P. Angelov and D. P. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 484–498, 2004.
- [14] P. Angelov and X. Gu, "Autonomous learning multi-model classifier of 0-order (almmo-0)," in *2017 Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, 2017, pp. 1–7.
- [15] P. P. Angelov and X. Gu, "Deep rule-based classifier with human-level performance and characteristics," *Information Sciences*, vol. 463–464, pp. 196–213, 2018.
- [16] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.
- [17] P. Angelov, *Autonomous learning systems: from data streams to knowledge in real-time*. John Wiley & Sons, 2012.
- [18] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *SIAM review*, vol. 41, no. 4, pp. 637–676, 1999.
- [19] H. Lee-Kwang, Y.-S. Song, and K.-M. Lee, "Similarity measure between fuzzy sets and between elements," *Fuzzy Sets and Systems*, vol. 62, no. 3, pp. 291–293, 1994.
- [20] E. Soares, P. Angelov, D. Filev, B. Costa, M. Castro, and S. Nagesh Rao, "Explainable density-based approach for self-driving actions classification," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 469–474.
- [21] K. K. Dobbin and R. M. Simon, "Optimally splitting cases for training and testing high dimensional classifiers," *BMC medical genomics*, vol. 4, no. 1, p. 31, 2011.
- [22] P. Cunningham and S. J. Delany, "k-nearest neighbour classifiers," *Multiple Classifier Systems*, vol. 34, no. 8, pp. 1–17, 2007.
- [23] I. Rish et al., "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22. IBM New York, 2001, pp. 41–46.
- [24] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.
- [25] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.
- [26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [27] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
- [28] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in neural information processing systems*, 2018, pp. 6638–6648.
- [29] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in neural information processing systems*, 2015, pp. 2962–2970.