Using Deep Learning to Replace Domain Knowledge

Christian Lübben*, Marc-Oliver Pahl‡*, Mohammad Irfan Khan†
*Technical University of Munich, ‡IMT Atlantique, †Eurecom
* {luebben,pahl}@s2o.net.in.tum.de; ‡ marc-oliver.pahl@imt-atlantique.fr † khanm@eurecom.fr

***Abstract*—Complex problems like the prediction of future behavior of a system are usually solved by using domain knowledge. This knowledge comes with a certain expense which can be monetary costs or efforts to generate it. We want to decrease this cost while using state of the art machine learning and prediction methods. Our aim is to replace the domain knowledge and create a black-box solution that offers automatic reasoning and accurate predictions. Our guiding example is packet scheduling optimization in Vehicle to Vehicle (V2V) communication. Within the evaluation, we compare the prediction quality of a labour-intense whitebox approach with the presented fully-automated blackbox approach. To ease the measurement process we propose a framework design which allows easy exchange of predictors. The results show the successful design of our framework as well as superior accuracy of the black box approach.**

***Index Terms*—V2V, V2X, network traffic prediction, deep learning, ANN**

I. Introduction

A common problem of complex systems is predicting their future behavior. The intuitive way is asking an expert who uses his domain knowledge. Experts are not always available or do not exist for certain problems. To illustrate this, we use an autonomous driving Vehicle to Vehicle (V2V) communication scenario. An integrated car systems expert knows about sending conditions of network packets. He is needed for forecasting at the moment. This knowledge comes with expenses such as money and time or is closed-source. Therefore, we replace domain knowledge by automatic reasoning and forecasting. We compare this black-box approach to a knowledge-driven white-box approach to evaluate which accuracy we could achieve while using no knowledge about the problem domain. The question we answer is:

To which accuracy can domain knowledge be replaced by a black-box predictor that uses no domain-specific knowledge?

On the way to level 5 fully autonomous driving, many challenges need to be solved. An approach to allow automated decision making is collaborative sensing. As standardized by the European Telecommunications Standards Institute (ETSI), different message types are used for collaborative measurements and awareness of other vehicles and obstacles. These messages are sent on a shared medium via the Intelligent Transport Systems (ITS) Networking & Transport Layer [1]. Messages can have different priorities and QoS requirements that need to be fulfilled in a real time scenario [2]. However, the shared broadcast medium has a limited capacity as the ITS-G5 standard uses IEEE 802.11p as basis which itself uses CSMA/CA as medium access protocol [2]. Therefore, an intelligent scheduler for transmitting network packets is needed. It has to consider the respective QoS requirements as well as the channel load to choose the best fitting time slot for a message emission. This requires knowledge about the future channel load. Predicting this load again requires expert knowledge about the V2V domain. In our example the explicit trigger conditions listed in the respective ETSI specifications should allow an accurate estimation of packet emission. By using recent prediction algorithms and supervised deep learning we assume this knowledge to be unnecessary.

To evaluate our research question we compare two prediction models. One model is based on domain specific knowledge (white-box approach, WB) using the information from the ETSI specifications. The other one follows a deep learning (black-box, BB) approach where no domain specific knowledge is involved. Both are then used to predict future packet emission within a real-world scenario. These predictions are compared to the number of actual sent packets to evaluate the prediction performance. Our scenario considers an intelligent vehicle that runs the intelligent scheduler and tries to predict channel load. All surrounding vehicles that are in transmission range, and thus visible to our vehicle, are in the following referred to as neighbors.

To test and evaluate the predictors we introduce a modularized framework that follows a divide and conquer approach. The incoming network traffic is subdivided into the respective packet types. Every type has its own predictor (WB & BB). This architecture guarantees easy replacement of predictors as well as extensibility to new packet types. In addition to the packet subdivision we do a traffic assignment for all neighboring vehicles. This guarantees the ability to predict traffic separately for each vehicle participating in the network.

Our *contributions* are:

- Deep learning versus domain expert approach comparison
- An easy to use, extensible prediction framework for network traffic
- Prediction models for black- and white-box approach

Section II introduces the tools used and the framework design. Section III presents our testdata and the used scenarios. Section IV gives a detailed view on our framework and the predictors. Section V discusses our results. Related works are

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

presented in section VI. Section VII answers our question based on the results and gives an outlook to future work.

II. Approach

In this paper we create a prediction framework that allows performing the measurements for evaluating our research question while also being extensible to other scenarios. Taking scenario challenges into account, we first form requirements for the framework. This is followed by an introduction to network traffic prediction and message types that have to be predicted. Finally we introduce our framework design.

A. Requirements

Within the evaluation as well as within the parameter tuning phase we need to replace prediction models. Therefore, models have to be exchangeable (R.1). While considering one message type at the beginning, our framework must be extensible to new message types that will be added to the V2V network traffic (R.2). To simplify the prediction task from global traffic to node specific traffic, overall network load is considered as sum of all participating node's traffic. This requires the interpretation and handling of node specific information (R.3). Besides the simplification of the prediction task the reuse of predictors is possible by using this approach. If only vehicles of the same kind are within the network, one predictor or one selection of predictors can be reused for each of them as nodes of a same kind should behave similar. In conclusion three requirements are set to our framework:

- R.1 Easy exchange of prediction models
- R.2 Extensibility to new message types
- R.3 Individual predictions for every network participant

B. Network traffic prediction

To enable QoS, our intelligent scheduler needs to predict the optimal timeslot for packet emission. We focus on collaborative awareness messages (CAM) as they make up the top most part of our traffic and thus have a high impact on the overall traffic prediction. Furthermore they are emitted on a structural pattern that can be learned by our BB approach and be used within the WB approach which wouldn't be possible at purely random emission. CAM were specified by the ETSI for the ITS standard and are used to inform nearby vehicles about the current location and motion parameters of the sender. Their periodical emission frequency is between 1Hz up to 10Hz and based on the fulfillment of three motion features [3]:

- Position change of 4 meters
- Speed change of 4 meters per second
- Direction angle change of 4 degrees

Fulfilling at least one of these conditions within a 100ms interval triggers a CAM emission. If the conditions are not fulfilled within 10 consecutive timeslots (1000ms) one CAM is sent to guarantee the minimal emission rate of 1Hz. Basis for the computation of the delta values are the conditions while sending the last CAM message.

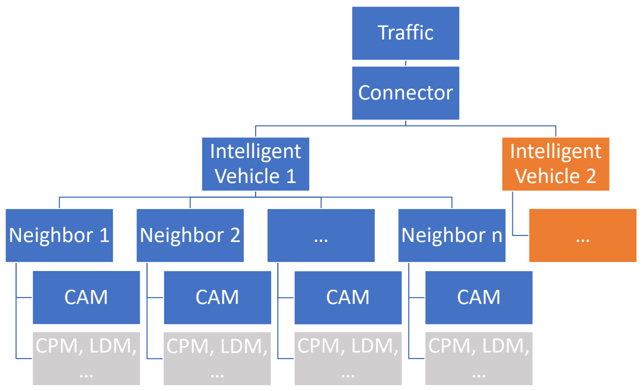

Fig. 1. Prediction framework design

C. Prediction framework design

The design of our framework is shown in figure 1. The input fed into the connector, is the raw network traffic. As stated before, we consider the global traffic as sum of all single nodes traffic in the network. This allows us on the one hand to do node specific predictions and on the other hand reduces complexity for the predictors as it is part of the proposed divide and conquer approach. In our scenario we focus on one intelligent vehicle that tries to predict the traffic of its surrounding vehicles. Based on the information contained in a single packet we are able to assign each message to its origin. Therefore, we can disassemble the traffic into node specific traffic. Each neighbor is represented by a "box". All received traffic belonging to one neighbor is forwarded to its box for performing the predictions there. In figure 1 we took Intelligent Vehicle 1 (IV1) as vehicle that is running our framework. For each neighbor IV1 creates a box. These boxes can be considered as logical representations of the neighbors. All prediction modules for one specific neighbor are bundled within there. A box thus can predict the whole traffic this neighbor emits (R.3). If we do not have any further information about the message types this neighbor emits, we can create one single predictor for all the emitted traffic. Since we know that we have specific message types, we can split up further. Each box therefore owns separate predictors for each message type. In our case this is one predictor for the CAM type. We assume that all vehicles implement the same specifications for emitting identical message types, as they are specified by the ETSI. Therefore, we can reuse the message predictors in every neighbor box. The specification of which message types should be considered within the boxes is done in the intelligent vehicles class. The predictors are then loaded once in the IV class and referenced from every box. If a prediction model needs to be updated, it can easily exchanged by loading the new predictor in the intelligent vehicle class (R.1). New models can also be added there together with a label for the corresponding message (R.2). The new predictor is then loaded and the list of predictors is extended by the new reference. In Figure 1 a list of additional packet types is indicated at the bottom. Our approach offers the flexibility to build a collection of predictors and choice which available predictors to combine within a scenario while reducing the prediction complexity.

III. Dataset

Our goal is to generate results that are applicable in real-world scenarios and measurements that reflect real-world behavior and complexity. Therefore, we took datasets that have been recorded in real environments. We use these datasets to train and test our models. Afterwards, we evaluate the prediction performance on a different subset. We ensure that no part of the datasets is used twice either in the learning, testing or evaluation process to avoid overfitting of the model. As we use real-world data, we also avoid unrealistic patterns that can occur in artificial generation. In particular we use two scenarios. One is a highway scenario recorded at Ingolstadt in Germany. The second one is a city scenario recorded in Bologna, Italy. We use both since they cover a lot of possible vehicle states. The highway dataset covers high speeds with minor changes. The city scenario includes low speeds with lots of changes due to acceleration and deceleration. With these datasets we are also able to do cross validation and evaluate how the predictors perform in the respective other scenario to gain information about the general applicability of the models. The size of the model creation dataset is 500k packets each for highway and city scenario. The dataset split was 80% for training and 20% for testing.

IV. Prediction framework

The prediction framework designed in II-C is now described in more depth. First we specify the technical setting. This is followed by the data preprocessing and a top-down specification of the framework ending with introducing the predictors.

A. Setting

The first thing we set is the size of the future time slice the predictions have to be done for. We use time slices of 100ms as it is the maximum emission frequency of CAM messages due to [3]. Our question to the predictor thus is:

How many packets will be emitted within the next 100ms?

Due to our divide and conquer approach this leads to the sub question of how many of our neighbors will emit a packet within the next time slice of 100ms? Whenever a packet is received the prediction process is initiated. Within this task a received packet gets forwarded to the corresponding neighbor box and further to the predictor of the received message type which predicts when the consecutive message will be sent. Input features are the current values sent in the received CAM message. The time in which a packet will be emitted is given in multiples of 100ms. The maximum value is 1000ms as it is the maximum time until a new CAM message is emitted [3]. This results in a range of 10 possible values for the prediction. These are the output classes for our multiclass prediction presented within the predictor introduction.

B. Supervised learning

While training our predictors we use supervised learning. This means that we use labeled data to train our models. As the needed information is supplied within the CAM messages, we do not have to manually label the data. Detailed information about the used labels as well as the used input features for the predictors is given in the following sections.

C. Preprocessing

Our prediction pipeline starts with preprocessing of the input data. At first, packets are assigned to the emitting neighbor and message type as introduced section II. Within the CAM predictor the payload data of the CAM message is processed further. A CAM message in our case includes:

SenderID, Timestamp, Position, HeadingAngle, MessageType

This information needs to be processed further to check the trigger conditions. To check if the position change exceeds 4m the position change between the current and last message ($\Delta Position$) is needed. In addition the speed within the current and last packet is needed to compute the speed change ($\Delta Speed$). This involves the $Speed$ value that relies on $\Delta Position$ and $\Delta Time$. Finally the change of the heading angle since the last message was sent ($\Delta Angle$) needs to be computed. By storing the information of the last received CAM message (t-1) and the current values (t) the needed values can be computed using the following equation:

$$\Delta Position = Position_t - Position_{t-1} \tag{1}$$
$$\Delta Time = Timestamp_t - Timestamp_{t-1} \tag{2}$$
$$Speed = |\Delta Position| / \Delta Time \tag{3}$$
$$\Delta Speed = Speed_t - Speed_{t-1} \tag{4}$$
$$\Delta Angle = Angle_t - Angle_{t-1} \tag{5}$$

D. White-box

To represent the emission conditions, we split the predictor into three parts. We use one predictor for each trigger condition. This design is shown in Figure 2, which is a detailed view of the CAM label from Figure 1. Instead of using the delta position we use the time dependent speed value as calculated in (3) to take into account the elapsed time since the last packet. This is needed since we need a time dependent prediction when the 4 meters will be passed. The idea by using the speed is to define speed intervals for each emission frequency. While driving more than 40m/s for example each 100ms the trigger condition of $\Delta Position > 4m$ will be fulfilled. This is again done to simplify the problem. We do not have to know the exact prediction value but only the CAM emission rate which is in a range of 10 possible frequencies. These frequencies are used as labels while training the models. They are encoded as an array of zeroes and a one for the affected interval also known as one-hot encoding. The translation of this one-hot encoded vector into the emission frequency and corresponding time periodicity is given in Table I. While entering the absolute speed values the speed predictor will predict the emission frequency for the future packet as one-hot encoded vector. For the conditions $\Delta Speed$ and $\Delta Angle$ we construct similar vectors. To include trends in the change of values we used time series prediction with one step back in time. After prediction, the resulting vectors are evaluated by taking their maximum index as all conditions are equally weighted. This determines the next emission timeslot. If the highest index within the three vectors is 5, the predicted emission frequency would be 2 Hz and the next packet emission is expected in 500ms (Table I).

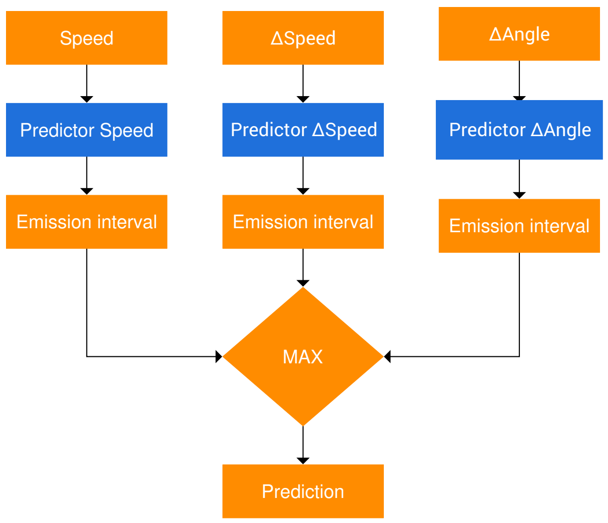

Fig. 2. Whitebox predictor

TABLE I
PREDICTION ENCODING

Index	0	1	2	3	4	5	6	7	8	9
Hz	1	1,112	1.25	1.429	1.667	2	2.5	3,334	5	10
Time	1000	900	800	700	600	500	400	300	200	100

E. Black-box

The black-box approach does not use any further domain knowledge. Within this approach we use the emission frequency of the packets as label. As our scheduler works with time slices of 100ms we again convert the learning labels to a range. While never exceeding 1000ms we consider everything else as outlier and fix the range to 10 possible values. The input is a subset of the available features that have been found within an automated grid search. These are $Speed$, $\Delta Speed$ and $\Delta Angle$. This design is displayed in Figure 3, which is the detailed view on the CAM label of Figure 1 for the black-box case. In contrast to the white-box design, we have only one predictor that has to learn the emission trigger conditions by itself. The output is again a one-hot encoded vector that can be decoded by using the information from Table I.

F. Model creation

Much effort has already been spent on general network traffic prediction, indicating promising applications of neural networks [9]. Due to their versatile application fields and the good results in previous applications we decided to use neural networks to implement our predictors. We tried to create the most simple model to avoid high computation overhead. While using large amounts of neurons one might achieve better results by risking that the models overfits and saves all model states. We started using a generic network approach to also keep the creation of the network simple. We started using 2 hidden layers with 10 neurons each by applying common guidelines regarding the number of hidden layer neurons in relation to input and output neuron number. After several tests we ended up with 50 Neurons within the hidden layers performing best. On an Intel Xeon E3-1265L V2 the training took around 1100 seconds per epoch. The resulting network

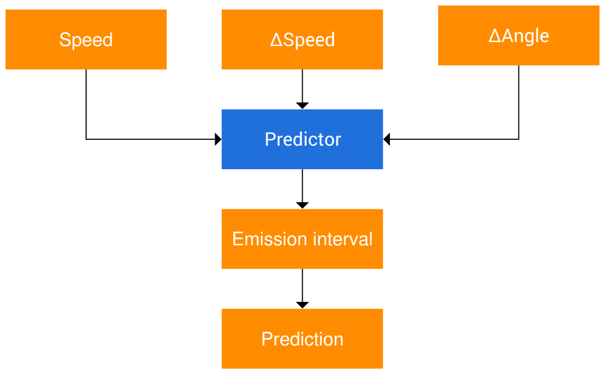

Fig. 3. Black-box predictor

starts with the input layer consisting of 5 input neurons that are fed with the 5 input values introduced in section IV-C. The two hidden layers each consist of 50 neurons. The connected output layer consists of 10 output neurons representing the one hot encoded output vector holding the classification results. We use categorical crossentropy loss function for our multiclass classification problem and rmsprop optimizer. We stopped training if the loss did not significantly improved anymore. While training the use of 3 epochs with a batch size of 1 performed good results. After training models for both scenarios they are loaded into our prediction framework. As stated in section II-C the predictors are loaded once within the intelligent vehicle class. For each received packet the predictor will classify the duration till the consecutive packet is expected. Each prediction run consists of:

- Make predictions for each received packet and save the expected time of arrival (ETA) within the neighbor box
- Collect all ETA that fall into the current prediction timeframe (next 100ms) from all neighboring boxes

Thus it is known how many packets are expected within the next timeframe in general and which neighbors are expected to send a packet. This answers our question to the predictor and implements R.3.

However, what happens if a packet gets lost and is retransmitted? As reasons for retransmissions can be manifold, we do not cover this case at the moment. Since we are not entirely sure if the expected time of arrival is the real time of arrival, we cannot be sure if a not received packet results in a retransmit or is just a prediction error. Another cause for not receiving a packet can be vehicles that leave the observed area. As current considerations produce more overhead and error than real value for the scenario, we decided to neglect the case of retransmissions. Another question within our ad-hoc network is: How to detect new network participants? In a pure random setting we cannot forecast which vehicle will enter our observation area. However, when we consider the number of entering vehicles equals the number of leaving vehicles the total number of vehicles should be balanced. In addition it is only one packet that is not predicted. After the first packet the neighbor is known and considered within the predictions.

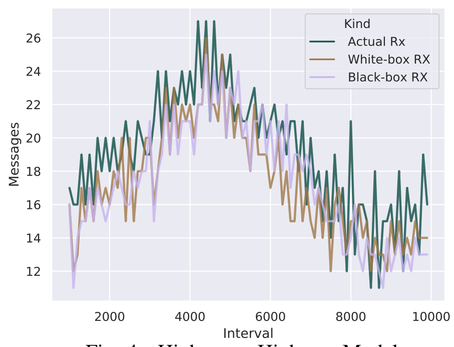

Fig. 4. Highway - Highway Model

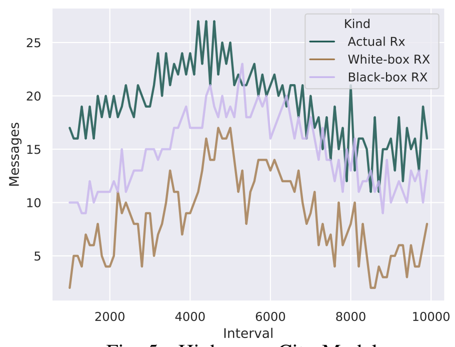

Fig. 5. Highway - City Model

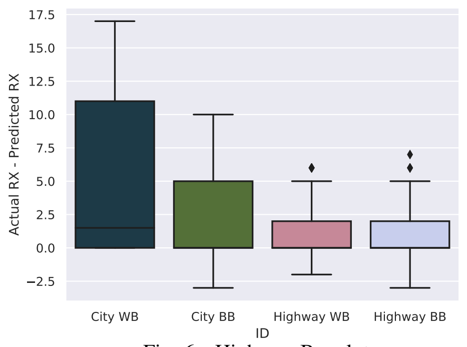

Fig. 6. Highway Boxplot

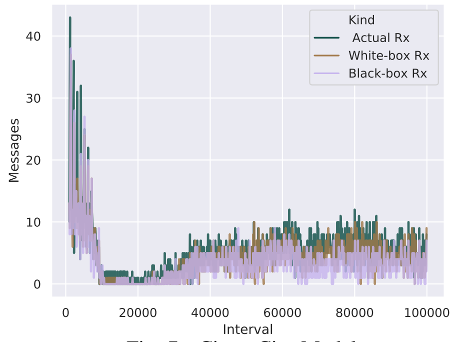

Fig. 7. City - City Model

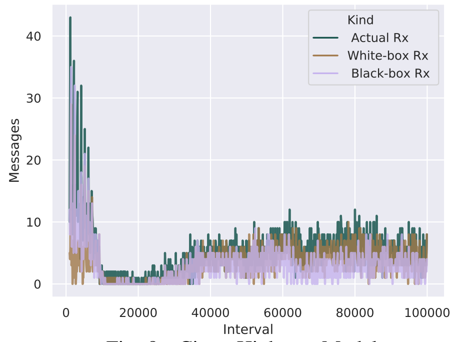

Fig. 8. City - Highway Model

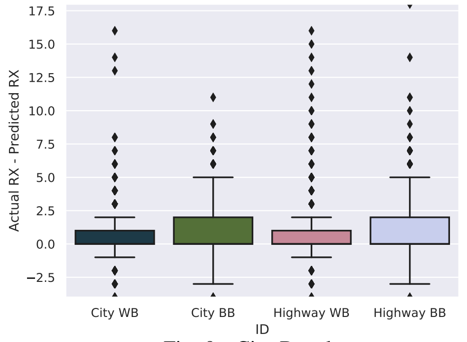

Fig. 9. City Boxplot

V. Evaluation

In the following we evaluate the results of our predictors within the two V2V scenarios. The evaluation is followed a comparison of both approaches. In total we perform eight different tests. First we use the trained models in their respective scenario. Second we cross check the models in the respective other scenario. The results are displayed in Figures 4, 5, 6 for the Highway scenario and in Figures 7, 8, 9 for the city scenario. The naming scheme of the graph first denotes the applied scenario and second the scenario where the used models have been trained in. The x-axis denotes the 100ms intervals for which the predictions are done. 1000 for example denotes the interval between 1000 and 1100. The y-axis denotes the number of packets within a 100ms interval. The three kinds of graphs within each Figure visualize the number of packets that have been actually received and have been forecasted by the WB and BB model. The Boxplots visualize the difference between actual received packets and the predicted amount of packets per predictor ($ActualRX - PredictedRX$). The mean absolute error (MAE) for each test run is printed in Table II. Within our evaluation we also check the requirements set to the framework. The requested easy exchange of prediction models (R.1) and extensibility to new message types (R.2) are realized through loading each model within the main class. By changing one code line models can be changed and by adding one line new models can be included. Individual predictions for every network participant are implemented by design (R.3).

TABLE II
Mean absolute error

Scenario	Highway WB	Highway BB	City WB	City BB
Highway	2.167	2.334	10.467	4.689
City	2.21	2.479	1.925	2.223

A. White-box

Boxplots (Fig. 6, 9) show that the WB predictors are able to predict packet emission with high accuracy resulting in a MAE of about 2.2 and 1.9 packets per interval. Even though there is a small error rate, we are still able to predict the general behavior to gain timeslots with high and less traffic (Fig. 4, 7). Since all predictors suffer from underprediction, a positive bias could be added. Within the cross check the city predictor performs significantly worse in the highway scenario (MAE>10) whereas the highway predictor only performs slightly worse in the city scenario (MAE 2.21).

B. Black-box

Within both scenarios the BB predictors suffer from light underprediction resulting in MAE of about 2.2 and 2.3 packets. Within the cross check, the highway predictor performs slightly worse in the city scenario with a MAE of almost 2.5 packets. Within the highway scenario, the city predictor performs significantly worse with a MAE of almost 4.7.

C. Comparison

In all scenarios the predictors tend to underpredict but also reflect the general trend of packet emission. Within the city scenario, the BB predictors perform slightly worse than their WB counterparts, resulting in higher MAE and wider variation (Fig. 9). Within the highway scenario, the respective WB and BB predictors perform almost equal whereas both city predictors differ significantly. Unexpected is the fact that the City BB predictor performs significantly better within the cross check than the WB predictor with a half as high MAE (4.7 vs. 10.5) and considerable less variation (Fig. 6). Comparing the WB and BB approaches shows that further knowledge of the problem domain results in a higher prediction accuracy in most cases. However, this is no surprise as it is to be expected. The question is, if the additional effort spent on implementing the domain knowledge is worth the effort. Compared to our black-box model we use three instead of one predictor, representing every trigger condition. This invokes more computation power and resource usage. As the prediction performance is in three cases almost the same and in one case significantly worse, at least in our guiding example, the additional effort and knowledge offer no sufficient gain in accuracy.

VI. Related work

[4] presents a design for a reusable machine learning service that offers easy extensibility and reusability. Like in this work, different predictors to validate the approach are used. However, the focus lied in simplifying the configuration and usage of ML on a usability scale instead of simplifying the whole model creation process. [5] discusses the usage of machine learning for vehicular networks. They give an overview and an example of a deep learning approach used in a vehicular network while proposing LSTM and RNN as potential improvements. The LSTM based approach for network traffic prediction that was presented in [6] is the basis of this work. It was shown that a RNN based approach can be used to predict future network traffic. However, there was no investigation of the invested domain knowledge like it is done in this work. As found out in this work, we could substitute the domain knowledge used there by a deep learning approach. [8] proposed a deep learning based algorithm for traffic prediction within a highway scenario which was later applied to a more general domain within the almost identical work by [7]. However, their approach targets on prediction traffic as a whole. We try to simplify the prediction problem by subdivision into network participant and packet type which is automated and easily possible through the included information provided by the transmitted packets. In [10] Rudin argues in favor of interpretable ML models. She claims that error and behavior prediction of BB models cannot be foreseen. She uses three examples where BB models lead to heavily wrong predictions. We support the statement that those models are not inherently better performing than interpretable models. However, we state that domain knowledge can be expensive or unavailable. Therefore, we claim BB models to be an easy way to implement predictors. In addition they can be more portable to other domains as they are not domain specific built. Our results show that the BB model does not necessarily outperform the domain specific solution but was easier to build while achieving good results.

VII. Conclusion & Future Work

In this work we compared two prediction approaches. One involving as much domain specific knowledge as possible and one that invests no domain specific knowledge. Both are validated and compared in two real-world scenarios. The evaluation showed that the benefit of using domain specific knowledge is minimal in both scenarios. In one case the black-box version even outperformed the white-box one. This leads us to an answer to our research question:

To which accuracy can domain knowledge be replaced by a black-box predictor that uses no domain-specific knowledge?

In our scenario that is based on a structured emission of packets we were able to replace domain knowledge through automatic reasoning with almost no loss in accuracy. In addition, the cross check reveals that the BB predictor seems to adapt better to new scenarios. However, the answer to this question also depends on the costs of the knowledge and the additional effort. This effort usually consists of time to invest, computing power or complexity to include the domain knowledge. In our case these costs support the replacement of our WB predictors. Since our guiding example was chosen as an exemplary prediction problem, our results should also be applicable in other domains that involve a non-random packet emission. Through the modular design the proposed framework can easily be equipped with predictors for new packet types or even be applied to new use cases. Further, it needs to be mentioned that often cases do not allow building white-box views as trigger conditions might not be known or are kept closed-source. Concerning those cases, we showed that black-box predictors combined with simplification of the prediction problem are able to produce good results.

There are still some open challenges including the extension of our framework to more packet types as well as the prediction of additional future timesteps. An aspect we did not cover in this work is the resource consumption and real time performance. These should be investigated on different hardware platforms, including special neural computation accelerators.

References

[1] European Telecommunications Standards Institute (ETSI), "ETSI EN 302 665", "Intelligent Transport Systems (ITS);Communications Architecture", Sept. 2010.

[2] European Telecommunications Standards Institute, "Final draft ETSI ES 202 663", "Intelligent Transport Systems (ITS);European profile standard for the physical and medium access control layer of Intelligent Transport Systems operatingin the 5 GHz frequency band", Nov. 2009.

[3] European Telecommunications Standards Institute (ETSI), "Final draft ETSI EN 302 637-2", "Intelligent Transport Systems (ITS);Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service", Sept. 2014.

[4] M.-O. Pahl and M. Loipfinger, "Machine learning as a reusable microservice", NOMS 2018, April 2018.

[5] H. Ye and L. Liang and G. Y. Li and J. Kim and L. Lu and M. Wu , "Machine Learning for Vehicular Networks", arXiv e-prints, Dec. 2017.

[6] M. I. Khan et al., "Deep Learning-aided Application Scheduler for Vehicular Safety Communication",arXiv:1901.08872, Jan. 2019.

[7] W. Wang et al., "A network traffic flow prediction with deep learning approach for large-scale metropolitan area network," NOMS 2018, 2018.

[8] Y. Lv et al., "Traffic Flow Prediction With Big Data: A Deep Learning Approach," IEEE Transactions on Intelligent Transportation Systems, vol.16, no.2, pp. 865-873, Apr. 2015.

[9] M. Joshi and T. H. Hadi, "A Review of Network Traffic Analysis and Prediction Techniques", 2015.

[10] C. Rudin, "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead", 2018.