Towards a Benchmark for Trajectory Prediction of Autonomous Vehicles

George Daoud* and Mohamed El-Darieby

Faculty of Engineering and Applied Science, Ontario Tech University, Oshawa, L1G 0C5, ON, Canada

George.Daoud@OntarioTechU.ca, Mohamed.El-Darieby@OntarioTechU.ca

*corresponding author

Abstract—The technology stack of connected and autonomous vehicles (CAV) consists of sensing, perception, motion prediction, and motion planning Layers. With much success, the sensing and perception layers have been developed. Recently, R&D activities on the prediction of vehicles trajectory have been attracting a lot of attention as it has the potential to increase safety for road users. Trajectory prediction is a significantly more difficult task because it involves capturing historical patterns of vehicle movements that requires an understanding and analysis of unstructured spatial and temporal data at the same time. Datasets that are used for this research are typically incomplete or not generic enough. Machine learning prediction models are developed in a bit of an ad hoc manner that they use various evaluation metrics. In this paper, we discuss the issues of such datasets, models, and evaluation metrics. We also present the requirements and initial high-level design of a benchmarking software framework that allows model users to search for and select already developed models, contributed by model developers, that process data collected by dataset contributors, and evaluated by the proposed framework. Further design and development of the proposed framework will ensue.

Keywords–Trajectory prediction; Benchmark; Bird's Eye View Models; Graph-based models

1. Introduction

Vehicular safety is an important area of research that receives increasing attention. Statistics show that in the US, the number of fatalities and injuries due to vehicle crashes changed from 36k and 2.74M in 2019 [1] to 38.7K and 2.28M in 2020 [2], respectively. CAV manufacturers and researchers have been focusing on (and are currently still) building CAV technology stacks that allow a vehicle to drive itself safely. While this was met with much success, some challenges still exist; especially after a few recently reported accidents.

The CAV technology stack consists of sensing, perception, motion prediction, and motion planning Layers. The sensing layer collects extensive data, estimated at a few terabytes/ second /CAV, from various sensors. The perception layer of the stack processes and augments such data to detect, classify and track road objects/ obstacles. The perception layer typically uses 2D & 3D bounding boxes (or polylines or even 3D Point Clouds) surrounding an obstacle, and segmentation of objects for classification purposes. Obstacles that can be identified include a) pedestrians; b) vehicles; and c) traffic signs, light signals, markings as well as driving conditions such as challenging lighting due to fog or night as well as lane lines and directions.

While the focus has been on developing the sensing and perception layers of that stack, research started to investigate the prediction of vehicle trajectory as it is increase driving safety for road users [3]. Trajectory prediction is a significantly more difficult task when compared to sensing and perception tasks. Trajectory prediction involves capturing historical patterns of vehicle movements in various real-life driving scenarios. In addition, trajectory prediction has a very dynamic domain that requires an understanding and analysis of unstructured spatial and temporal data at the same time. Examples of such unstructured data include a) there is no fixed or regular structure to represent data in one image (video frame) of data, b) the number of vehicles changes from one frame to another, and c) lane information and road constraints are difficult to be encoded.

At the current early stage of research in vehicular trajectory prediction, various models, datasets and evaluation metrics have already been proposed (albeit in an ad hoc manner) in the literature. We see a need for providing some structure for such research efforts. For example, the authors expect it to be very difficult to evaluate the performance of a trajectory prediction model using different datasets. This is because published datasets are not generic (designed for specific scenarios), some other datasets are not complete (missing fundamental information) and many datasets do not follow a standardized/ unified schema. These datasets were collected and curated by different organizations, in various contexts and for various purposes. Also, the required computational time and resources for the prediction process is a fundamental metric for real-life CAV that is not reported, to the best of our knowledge, by many (if not all) proposed prediction models. This renders current proposals not practical because predictions are typically performed locally within a CAV in real time to forecast the movements of surrounding objects and plan their motion actions safely.

This paper provides a first step towards defining the requirements for (and high-level design of) a benchmark software architecture to manage and compare various datasets and models for CAV trajectory predictions. We start by reviewing current research proposals and models and identifying their characteristics. In Section 3, we propose a set of requirements for the benchmarks. This is followed by a proposal for implementing the benchmark. We then present the conclusions.

2. The need for Benchmark

The authors recognize the need for standardizing a benchmark for CAV trajectory prediction to mitigate issues with datasets, models, and evaluation metrics as we describe in this section.

2.1. Issues with Datasets:

We list three problems with publicly available datasets:

1) *Not generic datasets*: where the data was collected for specific driving scenarios of a particular environment (e.g., for highways and not city streets). In addition, some datasets were collected for a specific geographic map that hinders general reuse into other models. In this work, we identify Predictable Driving Actions (PDA) as driving actions worthy of prediction. (In contrast, trajectory prediction is not needed for a vehicle travelling on crowded, with no potential for changing lanes, highway with no near-by ramps. It can be generally expected that such vehicles will not be taking a driving action other than driving along). Examples of PDAs include:
   - an intersection: driving straight forward through, turning left, turning right, U-turn
   - On a highway: on-ramping, off-ramping, change lane.
2) *Non-uniform datasets*: Collected datasets were collected by different organizations, for different purposes, within different contexts and with various data collection/ acquisition system and sensors (stereo cameras, LiDAR, Radar, Sonar, IMF, GPS, thermal imaging, and car internal CAN bus). Examples include CAV-generated datasets such as Lyft [4], Waymo [5], Argoverse 2 [6], NuScenes [7], and KITTI [8]. This leaves no room for uniformity in published datasets. For example, Lyft published its real-life dataset [4] collected in 2020 through driving 20 AVs in urban regions in California, USA. Such datasets are typically decomposed into several frames. Each frame captures the position of many vehicles and other types of road agents. Due to vehicular movements, not all agents appear in one frame. A set of frames that include all agents under consideration is called a scene. Raw data is preprocessed to recognize surrounding objects and determine their relative position and dimensions. Therefore, the dataset tracks many types of road agents like vehicles, pedestrians, and cyclists and covers many driving scenarios. The dataset also includes map information. Compare this to earlier non-CAV-generated datasets generated from stationary sensors pre-installed on the road; for example, NGSIM [9] that was collected from cameras fixed over a tall building and pointed over different segments of the Highways considered. Such videos are processed to identify (only) vehicles and track their relative positions over different video frames and are transformed to the same coordinates. Other efforts used drones to collect such information, for example, HighD [10], inD [11], rounD [12], and exiD [13], while others use Bluetooth devices [14] to form their dataset.
3) *Incomplete datasets*: the authors are not knowledgeable of a dataset that contained complete information. Examples of missing data include:
   - the location of a certain vehicle in different frames within one scene. In some datasets, this can be deduced from raw data; however, this leaves some level of inconsistency that might hinder evaluating results in a standard manner.
   - the direction of movement of vehicles.
   - map road, lane, and traffic signs data, which either not found in datasets or, at best, are inconsistent and semi-structured. This makes it very difficult to represent, link and process such data.

2.2. Issues with evaluation metrics:

The prediction models proposed show differences in evaluation strategies that hinder benchmarking. For example, while some research evaluates the prediction for every (future) second, other research evaluates performance only at the end of the prediction horizon (after a few seconds). Model evaluation strategies can be categorized into three strategies:

1) **Prediction Errors**: which compares predicted trajectory to actual trajectory at two levels:
   - Per-agent prediction error: the Displacement Error (DE) is defined as the Euclidean distance between the actual, $x_t^{(i)}$, and the predicted position , $\hat{x}_t^{(i)}$, of a vehicle as shown in equation (1). Two other error criteria are formed by decomposing the DE into two components in the direction of the lane, v_l, (named along-track) and in the direction perpendicular to it,$v_l^{\perp}$, (cross-track [15]) which are illustrated by (2) and (3), respectively.
   - Per-dataset prediction error: the following metrics, based on root mean squared error, are used to evaluate the prediction model over the whole dataset:
     - Average displacement error (ADE), given by (4), averages the DE over all the prediction horizons (t_f) for all prediction examples (n).
     - Final displacement error (FDE) is calculated using (5) by taking the mean of the DE evaluated at the end of the prediction horizon.
     - Other metrics include minFDE and minADE which are the minimum of ADE and FDE values calculated for each scene. While RF is defined as the ratio of avgFDE to minFDE [16].

$$DE^{(i)}(t) = \|\hat{x}_t^{(i)} - x_t^{(i)})\|_2 \tag{1}$$

$$AT^{(i)}(t) = \|v_l.(\hat{x}_t^{(i)} - x_t^{(i)}))\|_2 \tag{2}$$

$$CT^{(i)}(t) = \|v_l^{\perp}.(\hat{x}_t^{(i)} - x_t^{(i)}))\|_2 \tag{3}$$

$$ADE = \frac{\sum_i \sum_{t=1}^{t_f} DE^{(i)}(t)}{nt_f} \tag{4}$$

$$FDE = \frac{\sum_i DE^{(i)}(t_f)}{n} \tag{5}$$

2) **Prediction Reliability**: Others evaluate predictions models by evaluating the safety and the reliability of prediction

results (for example, does the prediction trajectory lie inside the drivable areas?). Map and lane information are mandatory for the application of these metrics. Zhang et al. [17] include two of those metrics:

- *Drivable area occupancy (DAO)* that the ratio of the number of map pixels in the predicted trajectory that lies within drivable areas to that of the whole trajectory.
- *Drivable area count (DAC)* is the ratio of the number of trajectory predictions that are totally within drivable areas to the total number predicted from the dataset.

3) **Prediction Performance**: AVs are real-life critical systems that have stringent time requirements for algorithms that control the movement of CAV on the road. It is mandatory to include computational (e.g., time and resources) performance and requirements in model evaluation.

2.3. Differences in Models:

In this section, we categorize prediction into two categories and discuss the need for benchmarking models.

2.3..1. BEV models:

Bird's Eye View (BEV) models extract images (picture frames) from CAV data and use them to train a Convolutional Neural Network (CNN) model to capture trajectory patterns. These models generate a semantic map that encodes vehicular location information into pixels of consecutive images. Other data such as travel direction of a lane and travel speed limit are digitized into color values. In spite of some loss of information in the generated map, this process converts variable dimensions of a dataset to a fixed dimension defined by the width and height of the generated image and the number of images in a (part of) scene.

Many research works investigated the use of CNN within BEV models. Mandal et al. [18] examine the performance of four ResNet and four EfficientNet models as the backbones of the CNN model. Huang et al. [19] proposed a convolutional-based Conditional variational autoencoder (CVAE) as the CNN architecture; the performance of which was found, by Jagadish et al. [20], to be worse than a customized residual network. Other approaches combine other ML architectures with the CNN in the same model as a transformer encoder followed by an RNN-based decoder [21], convolutional layers followed by an LSTM encoder-decoder network [22], or a convolutional LSTM-based encoder-decoder network [23].

To compensate for information loss and to achieve higher prediction accuracy, raw numeric data can be processed by other "ensembling" models as shown in Figure 1. Research work in [15], [24], [25] found "ensembling" to improve the prediction accuracy compared with non-ensembled models. This motivated Jagadish et al. [20] to propose another model (after their earlier work in [26]) in which two CVAEs are used one for the semantic maps and the other for numeric values. Other approaches, combine traditional models with a CNN model and use a third and final model to select between the two partial results [27], [28], [29].

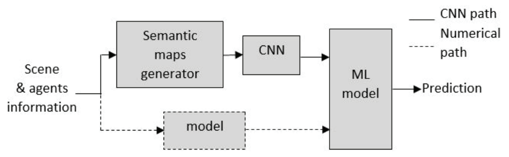


Figure 1: The general pipeline of BEV architecture

2.4. Graph-based models:

In these models, each frame is represented as a graph in which the nodes are the agents (vehicles), and the edges connect them to encode the "interaction" they impose on each other. Each node has a features vector that describes the agent like its dimension, position, and direction. Each scene is usually represented as a set of graphs. Using graphs allows the usage of raw data "as is" with no (or at least minimal) information loss (there is no need for discretization or digitization). This implies higher accuracy of input data to ML models which is expected to result in higher trajectory prediction accuracy. This also means a more compact input data size when compared to those of the BEV methods. However, the main disadvantage of graph-based methods is the difficulty of encoding "context information" of scene components such as lane geometry and traffic lights status in the graph. We classify graph models according to the order of applying spatial-temporal operations, as follows:

In Temporal-then-Spatial models, temporal information of each road agent is processed first to generate node embedding and features that represent the "interaction" between agents at a given instant of time. Then, a Graph Neural Network (or another deep learning technique) is used to predict the positions of agents in the following instant in the future. For example, Meng et al. [30] use an LSTM network to encode the temporal information into a 2D grid of tensor embeddings. Then, the grid is processed by a CNN to incorporate spatial information. Similarly, Yan et al. [31] use an LSTM-based encoder-decoder network followed by a spatial attention mechanism to encode the spatial interactions between agents into a single embedding that is used to predict the trajectory. Both transformers [32] and social pooling [33], [34], [35], [36] were used for spatial embedding. In [37], the graphs are stacked in a 3D matrix then a CNN is applied to the set of graphs. A revolutionary approach was proposed by Singh et al. [38] as they used positional encoding to make the model aware of "contextual" map information and applied a graph transformer to predict the future. Thus, this model performs better than other graph-based models when considering complex scenes.

In Spatial-then-Temporal models, each frame is processed individually to encode the spatial information including the interactions among agents. For example, A Graph-Convolutional Network (GCN) [39] is used to generate a spatial embedding for each frame, then, they are processed independently via a GRU-based encoder-decoder network. By representing the graph as a 2D matrix, a CNN is used for feature extraction before applying a decode-encoder network [40], [41]. Other approaches use an LSTM-based encoder-decoder for final temporal processing [41]. Another breakthrough toward a fully context-aware graph method is achieved by Vazquez et al. [42] in which the road components are encoded in a vectorized representation [43] to produce a map graph. The map graph and the vehicles-interaction graph are combined using an attention mechanism to produce a modified graph that attends to the important routes within the map before applying a GRU-based encoder-decoder network.

3. Proposed Benchmark

The benchmark has three main categories of users: a) Dataset contributor; B) Model Developer; and C) Model user. The proposed benchmark should fulfill the following requirements.

- Dataset management requirements
  - allowing a dataset contributor to add a new dataset to a repository
  - verify datasets
  - allow a model developer to search repo of datasets
  - provide a model developer with requested datasets formatted in training and testing data.
- Model management requirements
  - Allow a model developer to develop a model using a provided training dataset
  - allow a model developer to upload the developed model (and its functions) to a model repo
  - allow a model developer to upload the results of running the model on the testing dataset
  - evaluate and verify the results of an uploaded model and compare it to other models
  - allow model users to search various models and their performance metrics
  - allow model users to request and download a model with the corresponding dataset.

In Figure 2, we demonstrate a high-level architectural diagram of the benchmark software, which consists of a dataset and prediction model repositories. The benchmark software includes data and management processes that allow for adding, requesting, completing, and verifying a dataset as well as model management processes that allow for adding, searching for, and downloading to/from the repository.

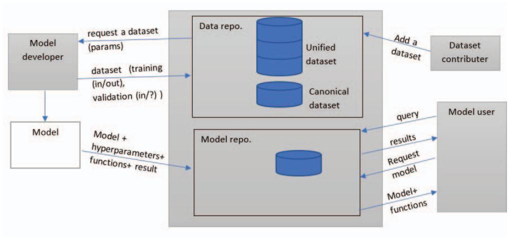


Figure 2: Architectural diagram of the benchmark

A fundamental requirement for the benchmark is to present a Unified dataset scheme that helps in designing a dataset repository. Figure 3 provides a preliminary design of such schema using an Entity-Relationship diagram. We designed the schema to comply with many published CAV-generated datasets in addition to complying with GIS software databases with API interface such as OpenStreetMap [44].

With the proposed schema, each dataset is divided into scenes, each of which represents a driving session. A scene has a starting_time, end_time, geographic scene_boundaries, and sampling_rate. These parameters allow for the identification of data frames that compose a scene. Each data frame contains information about agent vehicles that are identified and characterized using several attributes such as (position, agent_speed, and lane_ID: the lane used to travel). Agent vehicles that appear in a certain video frame share the same scene_ID and timestamp; this is enough information to identify the agent vehicle. Agents keep a list of frames (agent_previous_frame, and agent_next_frame) where they appear to maintain a history of their information. We identify agents with enough historical information as agents with potential for driving action predictions (APDAP); captured in the table APDAP in the schema. The table uses the parameter PDA to indicate which action is to be predicted for that particular agent. The table uses the variable maximum_history_length as a control parameter. Each agent has a maximum_prediction_horizon to identify the number of seconds in the future to predict the trajectory for the corresponding agent. If an agent appears in a scene for a very short time, There will be insufficient historical information on the agent, and consequently prediction of its trajectory is not possible. Stationary information such as the lanes, roads, and traffic lights are stored in separate tables with self-explanatory parameters. Traffic light status is captured in its table as it requires a timestamp parameter, through which it can be linked to a specific scene. This section discusses possible design and implementation techniques for the benchmark software.

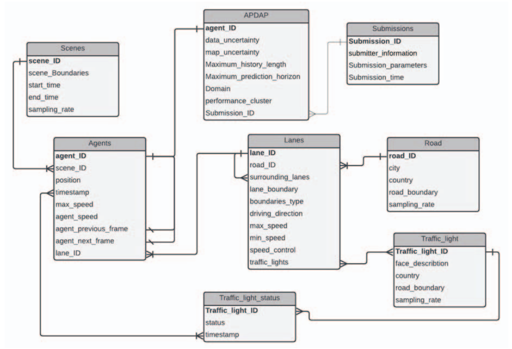


Figure 3: ER of the unified dataset

3.1. Search for and download a model

The benchmark should have a User Interface that allows a model user to search stored models (maybe using prediction output parameters or evaluation metrics). With the standardization of dataset, benchmark, and evaluation metrics, the system allows users to compare and choose the model that fits their needs and download a copy of a stored model. Researchers can take advantage of transfer learning to develop or update models. The model also can be deployed into a real-world application.

3.2. Benchmark Data Management Processes

The first use case for benchmark usage is when a dataset contributor would like to add a dataset to the benchmark repository. The step for this process is as follows:

- A verification process that assures conformity is applied

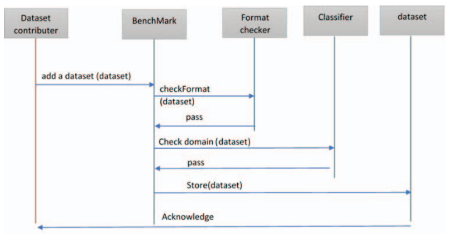


Figure 4: Sequence diagram of contributing new dataset.

- The new dataset is augmented to generate a more complete and more conformant synthetic dataset. The process is automated through many scripts and classifiers. For example, classifiers can detect missing driving scenarios and try to integrate data for them to ensure completeness. The APDAP table has data_uncertainty and map_uncertainty parameters to indicate the approximation of appended data.
- The quality of the dataset is evaluated. The dataset is used as input for baseline models to test dataset quality and performance. If the dataset produces an acceptable performance, it is added to the data repo.
- Mechanisms for undoing/removing a dataset from the repo should also be developed.

Another scenario of usage of the benchmark is when

1) Providing a data set to a model developer: When a dataset is requested by a model developer, the benchmark software is to provide a dataset that includes all driving scenarios of interest to the developer. To ensure uniformity, when data is sampled, the number of interesting frames describing the driving scenarios will be controlled.

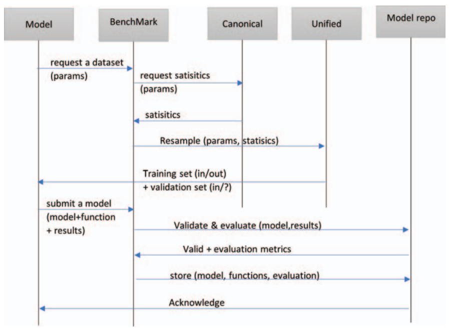

Model
BenchMark
Canonical
Unified
Model repo
request a dataset (params)
request satisitics (params)
satisitics
Resample (params, statisics)
Training set (in/out) + validation set (in/?)
submit a model (model+function + results)
Validate & evaluate (model,results)
Valid + evaluation metrics
store (model, functions, evaluation)
Acknowledge

Figure 5: Sequence diagram for model developing

2) Sampling a dataset: the goal of this process is to create a synthetic training dataset that meets the requirements of model developers (e.g., of a certain size and/ or with certain driving scenarios (domains) and/ or fulfilling a given set of hyperparameters). A synthetic dataset is sampled out of the benchmark data repository. This sampling is not done at random to satisfy the uniformity of driving scenarios. To achieve this, a canonical dataset and base prediction models are defined. The base model is applied to the dataset and a histogram of prediction errors is created for each driving scenario. The histograms define the canonical distribution of cases for each driving scenario. Then, this canonical distribution is used to sample the dataset to generate a statistically matching new dataset.

The benchmark software provides a model developer with a synthesized dataset to be used for developing the model. The dataset will be provided as two sets: 1) a training dataset with input data and output (labeling) data and 2) a testing dataset with only input data (Agent ID to predict its trajectory).

3.3. Benchmark Model Management Processes

1) A standard framework is needed for models to be deployed within the benchmark evaluation steps, used for transfer learning purposes, or deployed in a real-world application.
2) Adding a prediction model to the repository. A model developer submits a prediction model to save in the repository. The developer attaches the output of the prediction model. This output will be evaluated within the benchmark software to evaluate the performance of the model. If the model does not provide acceptable accuracy the model is not added to the repository.

The benchmark software provides needed helper functions that support frequent operators (create an image or a graph, data transformation) needed by models to execute.

To allow for injecting the model into the benchmark software (this is generally known as Inverse of Control, IoC, in software engineering), the model developer provides an implementation for a set of functions that follow an interface defined by the benchmark. Those functions are:

- Get_Prediction_Output_Parameters: returns a dictionary of model prediction parameters (e.g., history length, prediction horizon, sampling rate and type of model)
- Data_Preprocessing: preprocesses numeric data of each AgentID.
- Model_Input_Preparation: converts numerical data of each AgentID into a graph, map, or other customized representation.
- Model_Summary: returns a string describing the model implementation.
- Model_Load: loads parameters for transfer learning purposes.
- Model_Run: applies the model to a certain AgentID and generates a predicted trajectory.

3.4. Blind Evaluation of Model Performance

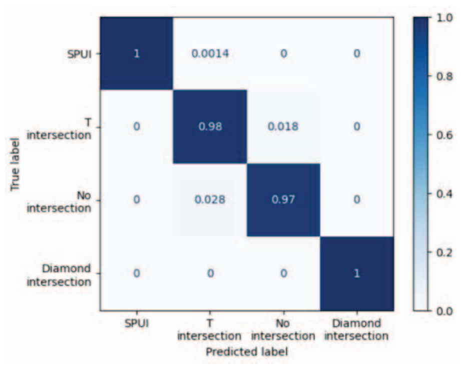
Evaluate samples of model output to evaluate model performance. The model developer uses the training dataset of the synthesized dataset to fine-tune the model. Then, the model is applied to the testing dataset and prediction output is produced. This output is compared to ground truth to be able to adjudicate the performance of the model. Output prediction results can be grouped and processed using, for example, AgentID. The benchmark can also calculate the computational time required to run the submitted model over standard hardware platforms. If the model passes those tests with acceptable quality, it is stored in a model repo with the corresponding functions, prediction parameters, and evaluation metrics.

4. Experiment and Results

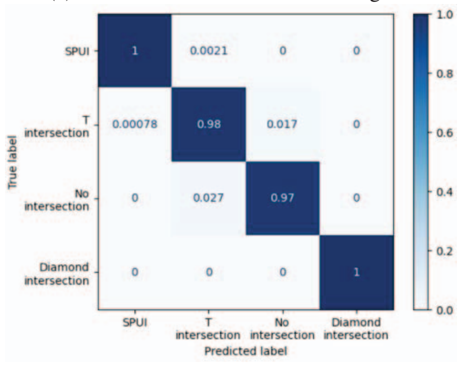
In this section, an examination is conducted to assess the feasibility of the proposed framework. Consequently, the framework's three primary challenges will be addressed. The initial challenge pertains to the absence of information within the dataset. This information can be readily obtained by utilizing the vehicle's location and the geographic coordinate system (specifically Latitude and Longitude), in conjunction with a geographic database such as OpenStreetMap, which provides map information. On the other hand, determining the missing domain of the intriguing cases presents a complex problem. Regrettably, numerous datasets, including the LYFT dataset, lack domain information. The second issue concerns the search and sampling of stored databases. The search process should not be restricted solely to the data already contained within the datasets; rather, it should possess flexibility and align with the objectives of the model designer. For instance, the user may describe a domain that is not encompassed by the stored datasets, and the framework should be capable of systematically examining all the relevant cases to identify those that satisfy the search criteria. Lastly, any newly submitted dataset should undergo validation to determine its acceptance or rejection. This validation process ought to be automated, requiring minimal human intervention.

The aforementioned challenges can be effectively addressed through the development of a machine learning (ML) model that operates on the standardized data representation provided by the framework. Specifically, the framework accommodates the utilization of Bird's Eye View (BEV) models, which offer a simplified approach and can be readily employed with the dataset. The proposed model, in particular, leverages an RGB image that serves as a representation of the map, along with a collection of binary images that depict the ego vehicle's location at the present and previous time frames. These inputs are utilized to generate four binary variables that correspond to four distinct domains, namely: Single-point urban interchange (SPUI), No intersection, T-intersection, and Diamond intersection. Notably, these four domains rely solely on the information derived from the map and the ego vehicle's location. However, if the targeted domains necessitate information pertaining to the scene's congestion or the interaction with surrounding vehicles, an additional set of images encompassing the surrounding vehicles can be incorporated as input data.

The input images undergo transformations including rotation, translation, and cropping, ensuring that the ego vehicle in the current time frame is consistently positioned and oriented, and that the images maintain a uniform size. The sampling rate for capturing the frames is set at 1 frame per second. Subsequently, all the transformed images are concatenated to form a unified input. This input is then fed into a ResNet-50 Neural Network (NN). The ResNet-50 NN is modified to accommodate the input layer's dimensions and number of channels, while the output layer is adjusted to generate four values that are subsequently normalized using a SoftMax activation function. The resulting values represent the probabilities associated with each domain. To establish a reliable ground truth, a manual labeling process is performed on 2000 cases from the LYFT dataset, encompassing instances from each domain. These labeled cases are shuffled and divided into two distinct sets: a training set comprising 70% of the data and a testing set comprising the remaining 30%. The training set is employed for optimizing the NN's parameters, while the testing set serves to determine the appropriate point for early stopping during the training process. Figure 6 illustrates the confusion matrix that delineates the performance of the NN in classifying the different domains, utilizing the testing dataset as well as the entire dataset.



(a) Confusion Matrix for the testing set



(b) Confusion Matrix for the whole data

Figure 6: Confusion Matrix for the domain prediction model

Based on the findings depicted in Figure 6, the model achieved accurate classification across all domains. The highest error rate, at less than 3%, was observed in the no-intersection domain. This can potentially be attributed to the similarity in driving patterns between vehicles immediately prior to entering an intersection and those proceeding straight through the intersection without turning. To mitigate false positive classifications, a threshold is employed, whereby classifications with probabilities below the threshold are either manually classified or disregarded.

Regarding the three primary challenges within the framework, similar models to the proposed approach can be utilized to address the issue of missing domain information in existing datasets. Additionally, these models can serve as a criterion for searching and sampling stored datasets. Instead of relying on query statements to search the dataset, users can employ binary ML models that determine which intriguing cases should be sampled and presented to the user. This same methodology can be applied to assess the integrity of newly provided datasets. By employing a set of canonical ML models, the output of the models can be compared with the data provided in the dataset, thereby evaluating its reliability.

5. Conclusions

At the current stage of R&D of vehicular trajectory prediction, various models, datasets and evaluation metrics have already been proposed (albeit in an ad hoc manner). This paper discussed issues with current state-of-the-art as it pertains to these datasets, models, and metrics. Published datasets are not generic, not complete and do not follow a standardized/unified schema as they were collected and curated by different organizations, in various contexts and for various purposes. While the model developed so far has covered a lot of exploration grounds, much more is needed. In particular, models that can process both spatial and temporal datasets that are intrinsic to the problem at hand. Even evaluation metrics need to be well understood and applied more rigorously.

In this paper, we proposed an initial vision, set of requirements, and high-level design of a benchmarking framework for such R&D. We explained how collected datasets can be verified and incorporated into the framework repository and how they can conform to a unified schema that we devised. We proposed ideas of how the framework can augment datasets with synthetic data. We also discussed how the framework enables model developers to request datasets to use in model development and how the framework prepares a dataset to serve developers' needs. We provided an initial design of how the framework can incorporate and run developed models and how it can be used to evaluate developed models and retain them if they show acceptable performance.

References

[1] "Traffic safety facts 2019: A compilation of motor vehicle crash data," National Center for Statistics and Analysis, Washington, DC, Tech. Rep., 2021. [Online]. Available: https://crashstats.nhtsa.dot.gov /api/public/viewpublication/813141

[2] T. Stewart, "Overview of motor vehicle crashes in 2020," National Highway Traffic Safety Administration, US, Tech. Rep., 2022. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public /Publication/813266

[3] A. R. Alozi and M. Hussein, "Evaluating the safety of autonomous vehicle–pedestrian interactions: An extreme value theory approach," *Analytic Methods in Accident Research*, vol. 35, p. 100230, Sep. 2022. [Online]. Available: https://doi.org/10.1016/j.amar.2022.100230

[4] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," 2020. [Online]. Available: https://arxiv.org/abs/2006.14480

[5] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov, "Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset," 2021. [Online]. Available: https://arxiv.org/abs/2104.10133

[6] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," 2023. [Online]. Available: https://arxiv.org/abs/2301.00493

[7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," 2019. [Online]. Available: https://arxiv.org/abs/1903.11027

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2012. [Online]. Available: https://doi.org/10.1109/cvpr.2012.6248074

[9] U.S. Department Of Transportation Federal Highway Administration, "Next generation simulation (ngsim) vehicle trajectories and supporting data," 2017. [Online]. Available: https://data.transportation.gov/d/8ect-6jqj

[10] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," 2018. [Online]. Available: https://arxiv.org/abs/1810.05642

[11] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein, "The inD dataset: A drone dataset of naturalistic road user trajectories at german intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Oct. 2020. [Online]. Available: https://doi.org/10.1109/iv47402.2020.9304839

[12] R. Krajewski, T. Moers, J. Bock, L. Vater, and L. Eckstein, "The rounD dataset: A drone dataset of road user trajectories at roundabouts

in germany," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Sep. 2020. [Online]. Available: https://doi.org/10.1109/itsc45102.2020.9294728

[13] T. Moers, L. Vater, R. Krajewski, J. Bock, A. Zlocki, and L. Eckstein, "The exiD dataset: A real-world trajectory dataset of highly interactive highway scenarios in germany," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Jun. 2022. [Online]. Available: https://doi.org/10.1109/iv51971.2022.9827305

[14] S. Choi, J. Kim, and H. Yeo, "Attention-based recurrent neural network for urban vehicle trajectory prediction," *Procedia Computer Science*, vol. 151, pp. 327–334, 2019. [Online]. Available: https://doi.org/10.1016/j.procs.2019.04.046

[15] A. Trabelsi, R. J. Beveridge, and N. Blanchard, "Motion prediction performance analysis for autonomous driving systems and the effects of tracking noise," 2021. [Online]. Available: https://arxiv.org/abs/2104.08368

[16] S. H. Park, G. Lee, M. Bhat, J. Seo, M. Kang, J. Francis, A. R. Jadhav, P. P. Liang, and L.-P. Morency, "Diverse and admissible trajectory forecasting through multimodal context understanding," 2020. [Online]. Available: https://arxiv.org/abs/2003.03212

[17] K. Zhang, C. Chang, W. Zhong, S. Li, Z. Li, and L. Li, "A systematic solution of human driving behavior modeling and simulation for automated vehicle studies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 944–21 958, Nov. 2022. [Online]. Available: https://doi.org/10.1109/tits.2022.3170329

[18] S. Mandal, S. Biswas, V. E. Balas, R. N. Shaw, and A. Ghosh, "Motion prediction for autonomous vehicles from lyft dataset using deep learning," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*. IEEE, Oct. 2020. [Online]. Available: https://doi.org/10.1109/iccca49541.2020.9250790

[19] X. Huang, S. G. McGill, J. A. DeCastro, L. Fletcher, J. J. Leonard, B. C. Williams, and G. Rosman, "Diversitygan: Diversity-aware vehicle motion prediction via latent semantic sampling," 2019. [Online]. Available: https://arxiv.org/abs/1911.12736

[20] L. M. D.N. Jagadish, A. Chauhan, "Deep learning techniques for autonomous vehicle path prediction," in *AAAI Workshop on AI for Urban Mobility. Vancouver, Canada*, 2021.

[21] M. Bhat, J. Francis, and J. Oh, "Trajformer: Trajectory prediction with local self-attentive contexts for autonomous driving," 2020. [Online]. Available: https://arxiv.org/abs/2011.14910

[22] X. Huang, G. Rosman, I. Gilitschenski, A. Jasour, S. G. McGill, J. J. Leonard, and B. C. Williams, "Hyper: Learned hybrid trajectory prediction via factored inference and adaptive sampling," 2021. [Online]. Available: https://arxiv.org/abs/2110.02344

[23] C. Kim, J.-K. Cho, Y. Jung, S.-W. Seo, and S.-W. Kim, "Action-conditioned traffic scene prediction for interactive planning," in *2022 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, Feb. 2022. [Online]. Available: https://doi.org/10.1109/iceic54506.2022.9748470

[24] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," 2018. [Online]. Available: https://arxiv.org/abs/1809.10732

[25] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," 2019. [Online]. Available: https://arxiv.org/abs/1911.10298

[26] D. N. Jagadish, A. Chauhan, and L. Mahto, "Autonomous vehicle path prediction using conditional variational autoencoder networks," in *Advances in Knowledge Discovery and Data Mining*. Springer International Publishing, 2021, pp. 129–139. [Online]. Available: https://doi.org/10.1007/978-3-030-75762-5_11

[27] C. Kim, H.-S. Yoon, S.-W. Seo, and S.-W. Kim, "STFP: Simultaneous traffic scene forecasting and planning for autonomous driving," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sep. 2021. [Online]. Available: https://doi.org/10.1109/iros51168.2021.9636255

[28] Z. Zhong, Y. Luo, and W. Liang, "STGM: Vehicle trajectory prediction based on generative model for spatial-temporal features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 785–18 793, Oct. 2022. [Online]. Available: https://doi.org/10.1109/tits.2022.3160648

[29] G. Kim, D. Kim, Y. Ahn, and K. Huh, "Hybrid approach for vehicle trajectory prediction using weighted integration of multiple models," *IEEE Access*, vol. 9, pp. 78 715–78 723, 2021. [Online]. Available: https://doi.org/10.1109/access.2021.3083918

[30] Q. Meng, B. Shang, Y. Liu, H. Guo, and X. Zhao, "Intelligent vehicles trajectory prediction with spatial and temporal attention mechanism," *IFAC-PapersOnLine*, vol. 54, no. 10, pp. 454–459, 2021. [Online]. Available: https://doi.org/10.1016/j.ifacol.2021.10.204

[31] J. Yan, Z. Peng, H. Yin, J. Wang, X. Wang, Y. Shen, W. Stechele, and D. Cremers, "Trajectory prediction for intelligent vehicles using spatial-attention mechanism," *IET Intelligent Transport Systems*, vol. 14, no. 13, pp. 1855–1863, Dec. 2020. [Online]. Available: https://doi.org/10.1049/iet-its.2020.0274

[32] L. Li, X. Sui, J. Lian, F. Yu, and Y. Zhou, "Vehicle interaction behavior prediction with self-attention," *Sensors*, vol. 22, no. 2, p. 429, Jan. 2022. [Online]. Available: https://doi.org/10.3390/s22020429

[33] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," 2018. [Online]. Available: https://arxiv.org/abs/1805.06771

[34] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao,

Y. Wang, and Y. N. Wu, “Multi-agent tensor fusion for contextual trajectory prediction,” 2019. [Online]. Available: https://arxiv.org/abs/1904.04776

[35] X. Xu, W. Liu, and L. Yu, “Trajectory prediction for heterogeneous traffic-agents using knowledge correction data-driven model,” *Information Sciences*, vol. 608, pp. 375–391, Aug. 2022. [Online]. Available: https://doi.org/10.1016/j.ins.2022.06.073

[36] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, “Pip: Planning-informed trajectory prediction for autonomous driving,” 2020. [Online]. Available: https://arxiv.org/abs/2003.11476

[37] X. Li, X. Ying, and M. C. Chuah, “GRIP: Graph-based interaction-aware trajectory prediction,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, Oct. 2019. [Online]. Available: https://doi.org/10.1109/itsc.2019.8917228

[38] D. Singh and R. Srivastava, “Multi-scale graph-transformer network for trajectory prediction of the autonomous vehicles,” *Intelligent Service Robotics*, vol. 15, no. 3, pp. 307–320, May 2022. [Online]. Available: https://doi.org/10.1007/s11370-022-00422-w

[39] J. An, W. Liu, Q. Liu, L. Guo, P. Ren, and T. Li, “DGInet: Dynamic graph and interaction-aware convolutional network for vehicle trajectory prediction,” *Neural Networks*, vol. 151, pp. 336–348, Jul. 2022. [Online]. Available: https://doi.org/10.1016/j.neunet.2022.03.038

[40] Z. Sheng, Y. Xu, S. Xue, and D. Li, “Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 654–17 665, Oct. 2022. [Online]. Available: https://doi.org/10.1109/tits.2022.3155749

[41] C. Ju, Z. Wang, C. Long, X. Zhang, and D. E. Chang, “Interaction-aware kalman neural networks for trajectory prediction,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Oct. 2020. [Online]. Available: https://doi.org/10.1109/iv47402.2020.9304764

[42] J. L. Vazquez, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool, “Deep interactive motion prediction and planning: Playing games with motion prediction models,” 2022. [Online]. Available: https://arxiv.org/abs/2204.02392

[43] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, “VectorNet: Encoding HD maps and agent dynamics from vectorized representation,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2020. [Online]. Available: https://doi.org/10.1109/cvpr42600.2020.01154

[44] J. J. Arsanjani, A. Zipf, P. Mooney, and M. Helbich, Eds., *OpenStreetMap in GIScience*. Springer International Publishing, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-14280-7