2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Action Sequence Predictions of Vehicles in Urban Environments using Map and Social Context

Jan-Nico Zaech[1] Dengxin Dai[1] Alexander Liniger[1] Luc Van Gool[1,2]
{zaechj,dai,alex.liniger,vangool}@vision.ee.ethz.ch

***Abstract*— This work studies the problem of predicting the sequence of future actions for surrounding vehicles in real-world driving scenarios. To this aim, we make three main contributions. The first contribution is an automatic method to convert the trajectories recorded in real-world driving scenarios to action sequences with the help of HD maps. The method enables automatic dataset creation for this task from large-scale driving data. Our second contribution lies in applying the method to the well-known traffic agent tracking and prediction dataset Argoverse, resulting in 228,000 action sequences. Additionally, 2,245 action sequences were manually annotated for testing. The third contribution is to propose a novel action sequence prediction method by integrating past positions and velocities of the traffic agents, map information and social context into a single end-to-end trainable neural network. Our experiments prove the merit of the data creation method and the value of the created dataset – prediction performance improves consistently with the size of the dataset and shows that our action prediction method outperforms comparing models.**

I. INTRODUCTION

Autonomous driving is expected to fundamentally change our understanding of mobility and give us safer and more efficient roads. One fundamental building block to achieve this, is the ability to predict future actions of other road users. Only if one is able to accurately predict the potentially multi-modal future, collisions can be avoided. However, predicting future actions and trajectories of other road users requires a comprehensive understanding of traffic scenarios. This includes understanding the static and dynamic environment, as well as the traffic rules and the unwritten rules that govern how road user interact with each other.

The most common approach in this research direction is to directly predict trajectories of other vehicles. While this is intuitive, allows for fully automatic data collection with today's test vehicles and yields a good representation for decision making and planning algorithms, it does not consider well that humans learn driving as a sequence of actions and also interpret driving scenarios that way. Furthermore, many traffic rules are defined as high-level representations using driving maneuvers. Thus an autonomous driving system requires emphasis on anticipating high-level actions of surrounding vehicles to better plan its own actions and to be more interpretable to humans.

To overcome the aforementioned challenges, we propose to state the problem of predicting traffic agents as the task of predicting action sequences and create a large-scale action prediction dataset based on real driving data. To circumvent the expenditure for manual annotations, we design an automatic method to convert trajectories of real-world traffic agents into action sequences. We apply the method to the large-scale dataset Argoverse [1] and compile a new action prediction dataset. The dataset contains 228,000 action sequences and features five distinct driving actions: *cruise* (↑, c), *turn left* (↖, tl), *turn right* (↗, tr), *lane change left* (↖, ll) and *lane change right* (↗, lr) that describe normal vehicle operation. For testing, 2,245 trajectories from the validation set have been manually annotated. The dataset allows us to train and compare models in a quantitative way in terms of their ability to predict a sequence of future actions. To our best knowledge, this is the largest dataset for action prediction of traffic agents derived from publicly available data. To facilitate the research in this direction, the data of this work will be made publicly available.

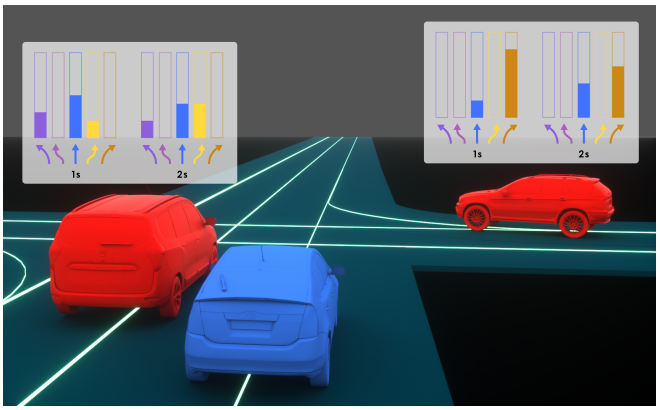

Fig. 1: Concept of the proposed method: Based on trajectory and map information, high level action sequences are predicted for surrounding vehicles. This representation is well interpretable and can potentially facilitate downstream planning tasks.

We further propose an end-to-end trainable neural network that uses the past positions and velocities of the traffic agents, the map information and the social context to predict the future actions of the traffic agents. To fully leverage the power of convolutional neural networks (CNNs), we encode the information into a rendered image with multiple channels and use 2D and 3D CNN modules for prediction. As future actions are often uncertain and multimodal, our model outputs a probabilistic distribution of all plausible actions which can facilitate downstream planning. Fig. 1 showcases

[1]Computer Vision Laboratory, ETH Zurich, Switzerland
[2]Dept. of Electrical Engineering ESAT, KU Leuven, Belgium

the concept of our action prediction approach.

II. Related Work

Behaviour prediction formulated as trajectory forecasting for humans as well as for vehicles has been extensively studied.

The line of research closest to ours focuses on predicting high level intention of traffic agents. One group of work in driver action prediction concentrates on the ego vehicle where rich information about the state is available. Morris et al. [2] use rich sensor data including radar, lane marking detection and a head tracking camera to predict lane changes in a highway driving scenario. Jain et al. [3] extend this approach to a larger set of maneuvers and base their method on a video of the driver, maps, vehicle dynamics and an outside view in more diverse environments. In [4] surround video and map renderings are used to predict yaw and acceleration in an end-to-end framework. [5] forecast lane-changes of other traffic agents in highway scenarios and analyze the challenge of heavily imbalanced data in this context. In more challenging urban environments, [6] classify driving actions at structured four way intersections with an LSTM based approach.

Early approaches to trajectories prediction, combine maneuver recognition with parametric motion models for each maneuver. Laugier et al. [7] use a Hidden Markov Model (HMM) with access to high level information such as distance to lane borders, signaling light status or proximity to an intersection to recognize behaviour of traffic agents. Trajectories are then sampled from a Gaussian process and used for evaluating collision risks. The same task is approached in [8] by detecting maneuvers with a Bayesian network. Houenou et al. [9] propose a heuristic maneuver detection module with full environment knowledge including each vehicle's acceleration and yaw angle, together with an analytic description of trajectory sets. By using a variational Gaussian Mixture Model, Deo et al. [10], [11] implement probabilistic trajectory prediction for highway scenarios in the combined maneuver and trajectory prediction framework. Recently, [12] used a neural network based approach to combine intention and trajectory prediction with dynamic HD maps and LIDAR information.

A second group of trajectory prediction algorithms directly approaches the task without intermediate state representations. Lee et al. [13] propose an end-to-end trainable recurrent neural network structure that includes scene context and samples multiple trajectories to capture the multi-modal nature of trajectory prediction. In [14], a wide range of output representations are evaluated in combination with a fully convolutional encoder structure. By defining a graph structure, [15] explicitly models the relation between multiple traffic agents and uses an LSTM-based encoder-decoder to predict trajectories. Closely related to trajectory prediction, researchers at Waymo [16] learn a driving policy using rich maps and employ data augmentation to train robust models.

In the context of pedestrian prediction and tracking, a central challenge is modeling of interactions. Pellegrini et al. [17] show that social interactions and scene knowledge can boost tracking performance. [18], [19] predict trajectories based on past ego and social trajectory observations, while matching not only trajectories but also distributions. Sadeghian et al. [20] include world context using a top down view and add attention to select the parts of the environment that are important.

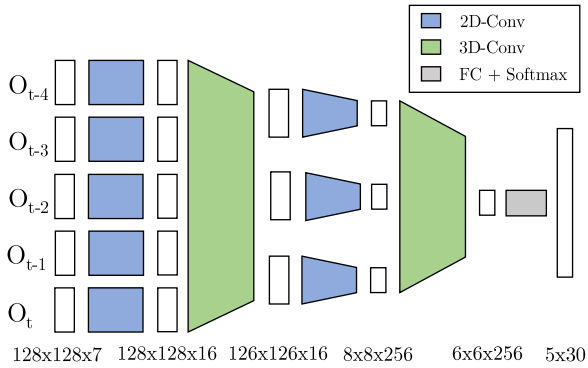

Fig. 2: VGG inspired network architecture, with 2D convolutions to extract spatial features and 3D convolutions for early and late temporal fusion.

III. Method

First, our method introduces an interpretable representation of traffic agents by forecasting high level action sequences of a single traffic agent. Second, our method is completely learning based and uses a sequence of map, agent, as well as social information (other traffic agents) to predict the agent action sequence. This is done using a fully convolutional neural network with two-dimensional convolutions for computing spatial features and two three-dimensional layers for late and early fusion of temporal features. The networks head is a fully connected layer that predicts the complete action sequence of the traffic agent at once with a single forward pass through the network. Finally, to make this method applicable for large scale datasets, we introduce an automatic approach to generate action sequences from X-Y trajectory data and map information. Note that our method is *entity-centric* and only predicts the action sequence of one agent, however we consider other traffic agents during prediction.

A. Network Architecture

We use a VGG-inspired architecture for action sequence prediction, which is shown in Fig. 2. Like [14], early and late fusion of temporal features is performed with three-dimensional convolutions. To avoid overfitting on the training data, we use a smaller model compared to some fully convolutional trajectory prediction approaches [14].

B. Input and Output Representation

The input to our neural network consists of three components: target agent, map and social information. All the information comes as a tensors with spatial dimension $128 \times$

128 and 7 channels at every timestep. We consider this combined information as our observation at time t, which we denote as $\mathbf{O_t}$. More precisely, $\mathbf{O_t}$ is given as,

$$\mathbf{O_t} = (\mathbf{m}, \mathbf{1}_{target}, \mathbf{v}_{target}, \mathbf{1}_{other}, \mathbf{v}_{other})_t \ , \qquad (1)$$

where, all components are rendered frames, spanning $50\,\mathrm{m} \times 50\,\mathrm{m}$. All frames are centered at the target agent's last observed position and rotated towards its driving direction. Each observation $\mathbf{O_t}$ contains one layer for the rendered lane centerlines $\mathbf{m}$ which stays fixed for all time steps. As the map only consists of centerline information, it can easily be captured with current industry grade perception systems and is present in most rich maps. Still, it naturally extends to more extensive information that might be available from manual annotations or more advanced data acquisition systems. Additionally, in $\mathbf{O_t}$ the target agent is represented by three layers: $\mathbf{1}_{target,t}$, which is the indicator function representing the target agent's position by a one-hot encoded layer and two layers $\mathbf{v}_{target,t}$ representing the current velocity in global coordinates. The other agents are represented the same way, but with all agents jointly rendered into the three available channels.

Altogether, the input to the network are the last five observations,

$$\mathbf{O} = \left(\mathbf{O}_{t_{-4}}, \mathbf{O}_{t_{-3}}, \mathbf{O}_{t_{-2}}, \mathbf{O}_{t_{-1}}, \mathbf{O}_{t_0}\right) \ , \qquad (2)$$

where the observations are spread over the last two seconds, with $t \in (-2\,\mathrm{s}, -1.5\,\mathrm{s}, -1\,\mathrm{s}, -0.5\,\mathrm{s}, 0\,\mathrm{s})$.

To improve the performance and robustness of our method, data augmentation is used. More precisely, we rotate the network input randomly by θ, which is uniformly sampled from the range $-5^\circ \leq \theta \leq 5^\circ$. To perform the augmentation efficiently and without artifacts, all input data is stored in parametric form and rendered on-the-fly.

The output of the network is an independent probability distribution of the 5 action classes for 30 timesteps. With a sampling time of $100\,\mathrm{ms}$, this results in a prediction horizon of $3\,\mathrm{s}$. Note that the temporal relation and implicit dependence needs to be learned by the model from the training data.

C. Probabilistic Action Predictions

In contrast to trajectory forecasting methods, our approach directly returns probabilistic predictions, without any requirement for sampling multiple forecasts [13] or defining spatially discretized grid-maps, as done in [21], [14]. Furthermore, this also allows for transparent performance measures since action classes can easily be interpreted by humans. The performance evaluation of trajectory prediction methods on the other hand normally uses averaged displacements errors. Even thought this seems to be a transparent evaluation, due to imbalanced datasets, where following the current lane (our cruise action) is heavily over-represented, getting better displacement errors not necessarily implies that the method is better at forecasting the important corner cases. Note that the imbalance in the datasets can be massive, e.g. our dataset consists of 80% cruise states, however, there are other traffic environments where cruise trajectories can make up as much as 99% of the dataset [5]. This imbalance also explains that often simple constant velocity forecasting methods are competitive for short prediction horizons [14], even thought they lack any understanding of the traffic scenario.

By predicting action sequences, our method does not directly solve the imbalance in the dataset. However, our evaluation method is more fine-grained and focuses on complex scenarios that require to understand the traffic scene.

IV. Dataset

Many large scale datasets for visual tasks in autonomous driving such as image segmentation, object detection or depth estimation have been released in recent years and fueled the development of corresponding learning based methods [1], [22], [23], [24]. In contrast to this, modeling and prediction of human decision making in driving scenarios is heavily underrepresented, which can be attributed to the lack of data. Public datasets, while being suitable for vision tasks, fall short when approaching the task of modeling human driving behaviour in complex environments. Furthermore, most datasets directly intended for this purpose remain private [12], [14].

A. Argoverse

An exception is the Argoverse Trajectory Forecasting dataset [1], which contains approximately 325,000 automatically detected trajectories in an urban environment. Out of them, 245,000 cover 5s segments that can by used for our approach[1]. The dataset further provides basic semantic map information, including lane centerlines and the drivable area. In our work we augment this dataset by automatically annotating high-level actions such as lane changes and turns that are interpretable by humans and have the potential to boost low level tasks like trajectory forecasting [11]. Finally, we also automatically extract velocity information of the agents, which further helps our prediction model.

B. Map Information

As shown and discussed in Section II, using HD-map information can be fundamental for traffic agent forecasting. However, in this paper we show that HD-maps can also be used to automatically annotate data, or in our case generate high-level action sequences from trajectories. This avoids time and labor intensive manual labeling, and at approximately 245,000 trajectories in Argoverse, which corresponds to roughly 7.5M action annotations, this is the only viable approach.

One the one hand, relying on HD-maps for action sequence generation, is in our opinion not restrictive, since maps are regarded essential for safe operation of autonomous vehicles. On the other hand, for large scale datasets, annotating a semantic map with lane centerlines, which remains constant over time, scales favorably compared to annotating every time step of every agent recorded during driving. Furthermore, as current lane detection algorithms show an

[1] 80,000 trajectories are in the test set and only show 2s segments.

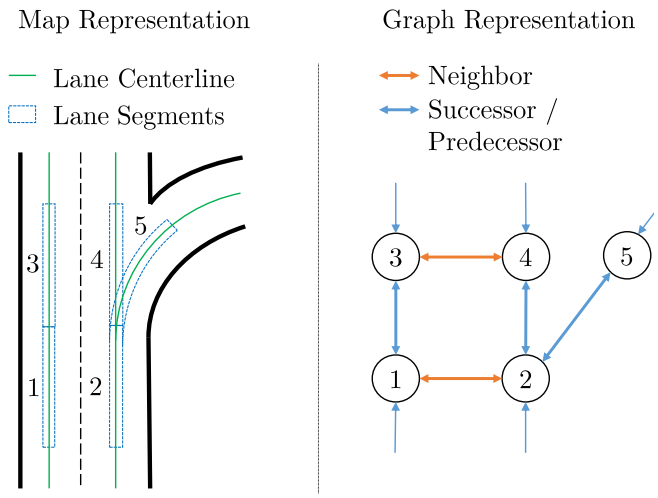

Fig. 3: Map represented as rendered centerlines and as graph-map. Each centerline corresponds to a node in the graph-map which are connected by edges labeling their semantic relation.

impressive performance in a wide range of practical scenarios, automatic extraction of local map information could be feasible to further automate the labeling process.

For the task of action sequence annotation, the semantic HD-map available in [1] can be represented as a graph where each node corresponds to a short sequence of line segments and edges represent the relation between the line segments, which is visualized in Fig. 3. Edges can have the labels successor, predecessor, and neighbor lane-segment. Nodes contain the geometric properties describing the lane segments and semantic information if the segments describe a turn (either left or right) or a lane driving straight forward.

C. Automatic Action Labeling

Action extraction from trajectories is performed in a three-stage pipeline described in the following paragraphs.

a) Trajectory Smoothing: As the trajectories are generated from automatically detected objects, all samples are noisy and need to be filtered in the first stage. For this purpose we use a bidirectional Kalman filter with a standard constant acceleration model, where the jerk is modelled as a noise input and we use the agent's noisy position as the measurement. Note that this filter does not only help to smooth the agent's position but also estimates the velocity of the agent.

b) Lane Assignment: In the second stage of the annotation pipeline, each sample from the trajectory is assigned to a node, which represents a lane-segment in the graph-map as shown in Fig. 3. Following the temporal order of samples, each trajectory induces a sequence of nodes that are visited. If the sequence of nodes follows a valid path through the graph, i.e. a path that only contains transitions between nodes that are connected by an edge, actions can be extracted from the graph. The property that only a valid path allows for the extraction of an action sequence form the base to design the node assignment algorithm. Using a purely geometric approach, where each sample is assigned to its nearest neighbor in an arbitrary measure, could lead to a large fraction of invalid trajectories. We thus propose to use a joint geometric and semantic lane assignment algorithm based on the Viterbi algorithm.

One can interpret a trajectory as the observation from a Hidden Markov Model (HMM) defined by the graph-map together with actions as latent variables. The graph-map induces a HMM, where the lane segment used by the agent at time step t corresponds to the hidden state X_t and it's position $Y_t \in \mathbb{R}^2$ is the symbol emitted by the state. The emission probability

$$P(Y_t|X_t = x_t) \tag{3}$$

captures the driver's behaviour of not perfectly following the centerline, uncertainties in the map and measurement noise. Using this viewpoint, the actions taken by the agent define the sequence of hidden states, or inversely: inferring the most likely sequence of hidden states from the observed trajectory is equivalent to finding the underlying action sequence. This task can be solved with the Viterbi algorithm.

To reduce computational complexity, only lane segments that are closer than a threshold of $5\,\mathrm{m}$ to the observed trajectory are included as hidden states in the HMM. Prior knowledge about the driving behaviour can be encoded in the transition matrix $\mathbf{A}$ between lane segments, which we populate using the edges of the graph-map that describe the relation between lane segments. For the possible transitions to a successor, predecessor or neighbor lane, values of 1.0, 0.5 and 0.3 are assigned respectively. For transitions between not connected lanes, a skew variable α is introduced. Setting $\alpha = 0$, yields state sequences that can always be annotated, given the map information is valid. This also includes trajectories that actually cannot be modelled by the 5 action classes used in this work, such as U-turns. Thus, we set $\alpha = 0.001$ which allows for transitions between unconnected lane segments, preventing the annotation of an action sequence for cases that cannot be annotated with the given set of actions. It is important to note that for the transition matrix $\mathbf{A}$ the sum of entries from a state does not necessarily sum up to 1 and thus, does not represent a traditional transition probability matrix. We use this representation to not penalize lane-segments that have multiple lanes connected to them in the graph-map, such that trajectories going through them do not come at a higher cost.

c) Action Extraction: With the most likely sequence of lane-segments determined by the Viterbi algorithm, actions can be extracted from the map information with a rule based system. Lane changes are labeled for transitions between lane segments marked as neighbors[2] and turns can directly be labeled from the turn annotation in the lane segment.

All actions are handled as non-singular events, e.g. the lane change state is annotated for all time steps between leaving

[2]Corner cases such as the direct change to the neighbor of a successor lane need to be modeled and handled adequately.

			Turn		Change	
Split	Total	Cruise	left	right	left	right
train	191,841	84.5%	7.6%	4.0%	2.1%	1.9%
val	36,471	87.5%	4.0%	3.2%	2.9%	2.4%
manual	2,245	87.2%	4.1%	2.7%	3.5%	2.6%

TABLE I: Label distribution of the annotated data.

the previous lane and stabilizing on the new lane. To enable this, the smoothed trajectory together with the lane geometry is used to extract the start and end point of all maneuvers.

D. Test Data and Ordered Action Sequences

For testing, 2245 trajectories from the validation set have been manually annotated. However, the annotator did not generate temporal action sequences, but what we call ordered action sequences. Whereas action sequences, as produced by our automatic labeling, contain an action for each time step, ordered action sequences only contain the order of actions. To make this clear, let us consider an simple left lane change example where the prediction horizon is six time steps. In our example the the action sequence is given by $\mathbf{a}_s = (\text{c,c,ll,ll,c,c})$. This action sequence would correspond to the following order action sequence $\mathbf{a}_{os} = (\text{c,ll,c})$, thus the exact temporal location of the lane change is lost, however, the gist of the maneuver is captured.

Annotating ordered action sequences is also significantly simpler and less error prune, compared to temporal action sequences. Thus, for our 2245 test trajectories, we have hand annotated ordered action sequences for the $3\,\text{s}$ prediction horizon. Note that to avoid bias, the annotator had no access the automatically generated labels.

E. Dataset Statistics

While the trajectories present in the Argoverse dataset are already filtered to show challenging behaviour, the extracted action data is still imbalanced, with the majority of the samples representing the cruise class as shown in Table I. While this mostly stems from the lower probability of encountering an active maneuver compared to just following a lane in cruise, it also reflects the fact that a turn or lane-change is usually followed by the traffic agent stabilizing in the cruise state.

Whereas generating the dataset statistics for our automatically generated action sequences in the training and validation set is based on the number of occurrences, the statistics for the ordered action sequences uses an adapted method. State proportions are estimated by counting states with the inverse number of total states annotated for the corresponding sequence, e.g. a sequence only annotated as cruise counts 100% towards cruise, while a sequence consisting of cruise and a following lane change counts with 50% towards both classes. In total, the state distribution for the train, validation, and test set is as shown in Table I.

V. Experiments

A. Training

Network parameters are optimized with the Adam optimizer and an initial learning rate of 10^{-4}. A step decay schedule with a stepsize of 10 epochs and a factor of 0.5 is used for adapting the learning rate. Dropout of 0.5 is used to reduce overfitting. All variations of the network are trained for 50 epochs.

To compensate for the heavily imbalanced dataset, weighted random sampling is used for dataloading. Weights are assigned based on the presence of actions anywhere in the prediction horizon: if a turn is present, the sample gets weighted with a factor of 3, if a lane-change is present, the weighting factor is set to 10. This does not fully compensate for the imbalance, but performed better than strictly weighting samples by their inverse probability.

Note that the positions and velocities of all traffic agents used in the input representation in Equation (1) are extracted from the noisy trajectory data with a bidirectional Kalman-filter, as described in Section IV-C. However, to ensure causality, the filter is only applied to the observed values.

B. Baseline Model

By proposing a new formulation of the traffic agent prediction task, no direct comparison to other methods is possible. However, trajectory forecasting methods are intended to predict a trajectory that represents the future actions of the agent. Therefore, it is possible to adapt a method originally designed for trajectory prediction to the new task. For our evaluation, a k-Nearest-Neighbor (k-NN) based method that was evaluated in [1][3] and outperformed all their tested deep learning models for multimodal prediction is adapted to our problem statement and used as a baseline-model for comparison. With the nearest neighbors in the training set, the automatically extracted action sequences can be used as a prediction for the task at hand. Given the predictions from k-NN, the frequency of class labels at each time step results in a prediction that matches the probabilistic form of our proposed approach. We set $k \in \{9, 50, 100\}$ to compare the influence of sampling multiple trajectories and select the best model for further comparison.

C. Evaluation Approach

1) Direct Evaluation of Predictions: Interpreting every time step as an independent classification problem, allows for the direct evaluation of the network's performance. As the data is heavily imbalanced, classification accuracy on the whole dataset does not provide sufficient insight. Therefore, we measure performance by representing every action class as a binary classification problem, which allows the calculation of the average precision (AP) score. AP for each class and their unweighted average denoted as mean AP across all five action classes are used as a performance measure. This direct evaluation approach jointly measures the performance of predicting the right action classes together with the

[3]arXiv:1911.02620 [v1] 6 Nov 2019

Method	Mean	Cruise	Turn left	Turn right	Change left	Change right
random	20.0	87.5	4.0	3.2	2.9	2.4
k-NN (9) [1]	32.5	93.5	32.9	26.3	5.4	4.2
k-NN (50) [1]	36.9	95.0	39.2	34.7	8.2	7.1
k-NN (100) [1]	37.4	95.3	39.7	35.4	8.6	8.0
ours	**61.4**	**97.8**	**67.9**	**68.4**	**33.5**	**39.4**

TABLE II: AP scores of the investigated methods on the automatically annotated validation set. Random sampling performance corresponds to dataset proportion for each class. All scores in %.

performance of predicting the right time step for a transition between two actions.

2) N-Most Likely Ordered Action Sequences: In addition to directly measuring the model's precision on the temporal action predictions, we evaluate our method by extracting the N-most likely ordered sequences of actions, i.e. sequences that are only defined by the order of actions, not their exact length or transition times.

The extraction of the N-most likely ordered action sequences, is sensible as the number of different actions in the 3 s prediction horizon is small. For our test set only 10 samples, corresponding to 0.45% of the manual annotations, where labeled as as sequence of more than two actions. Therefore, for extracting ordered action sequences from the temporal action sequence predictions, we only consider sequences with at most two actions, which makes the approach tractable.

When extracting ordered action sequences, the model of independent actions used during training needs to be taken into account. While this assumption is a good model for the probability of an agent performing action a_t at time step t, it is not suitable to approximate the probability of observing a full action sequence $\mathbf{a}_s = (a_0, ...a_T)$. By just using the product of the predicted probabilities

$$p_{ind} = \prod_t \hat{p}_t^{a_t}, \tag{4}$$

as an estimate for the sequence probability, the high temporal correlation of actions is neglected. We thus model the probability of a sequence with the two actions (a_{b1}, a_{b2}) by

$$p_b(a_{b1}, a_{b2}, t_s) = \min(\hat{p}_{t_0}^{a_{b1}}, ..., \hat{p}_{t_s}^{a_{b1}}) \min(\hat{p}_{t_s+1}^{a_{b2}}, ..., \hat{p}_T^{a_{b2}}), \tag{5}$$

where the transition happens after time step t_s leading to the blocks $t_0 \leq t \leq t_s$ of a_{b1} and $t_s < t \leq T$ of a_{b2}. Given that within a block no transition may happen, the transition probabilities between identical actions are 1. Therefore, the total probability of the first block can be modelled as the lowest predicted probability for a_{b1} within $t_0 \leq t \leq t_s$

$$\min(\hat{p}_{t_0}^{a_{b1}}, ..., \hat{p}_{t_s}^{a_{b1}}). \tag{6}$$

Note that the same holds for a_{b2} and the second block. Under the assumption that rare combinations, such as a left turn directly followed by a right turn, are assigned low probabilities by the network, the transition probabilities between the blocks are modeled as being equal for all action pairs (a_{b1}, a_{b2}).

Sequence	Ours Top1	Ours Top2	Ours Top3	k-NN (100) [1] Top1	k-NN (100) [1] Top2	k-NN (100) [1] Top3
total	82.5	89.8	93.0	81.2	86.4	88.6
c	94.1	96.2	97.9	99.6	99.8	99.9
ll, c	65.7	85.7	95.7	1.4	27.1	41.4
tl	63.3	69.4	73.5	14.3	18.4	26.5
lr, c	44.7	72.3	83.0	0.0	36.2	53.2
tl, c	25.0	65.0	87.5	32.5	95.0	97.5
tr, c	42.5	75.0	80.0	17.5	70.0	87.5
c, tl	12.1	69.7	75.8	0.0	42.4	69.7
c, tr	24.2	72.7	78.8	0.0	36.4	48.5
c, ll	4.3	52.2	73.9	0.0	17.4	26.1
ll	23.8	38.1	42.9	0.0	0.0	0.0
lr	11.1	22.2	22.2	0.0	0.0	0.0
tr	35.3	47.1	58.8	0.0	0.0	5.9
c, lr	6.3	75.0	75.0	0.0	6.3	6.3
tr, ll	14.3	21.4	42.9	0.0	0.0	0.0
tl, lr	40.0	50.0	60.0	0.0	0.0	10.0

TABLE III: Evaluation of the top-N ordered sequence predictions. The shown sequences are ordered by descending frequency and at least have 10 samples. All scores are accuracies in %.

The problem of finding the most likely ordered sequence of two actions can then be defined as finding (a_{b1}, a_{b2}, t_s) that maximize the product of the two block probabilities

$$p_{b,opt} = \max_{t_s, a_{b1}, a_{b2}} p_b(a_{b1}, a_{b2}, t_s). \tag{7}$$

By repeatedly searching for the most likely sequence while suppressing already extracted action pairs (a_{b1}, a_{b2}), the N-most likely ordered sequences can be extracted. We report total and per sequence top-N accuracy with $N \in \{1, 2, 3\}$ in Table III on manually annotated data. A sequence is rated as being detected if the groundtruth ordered state sequence is one of the top-N predictions. Trajectories that are annotated with sequences of length > 2 are always treated as being predicted incorrectly, when computing the total accuracy.

VI. Results

A. Direct Evaluation

Performance metrics for the direct evaluation of predictions for the k-NN baseline [1] and our proposed method are shown in Table II. The results affirm that increasing the number of sampled nearest neighbors by one magnitude (k=100) compared to the usual setting can improve its performance. Across all methods, the AP for the cruise state is on a high level, followed by turn actions. Lane changes are harder to predict with AP values at a much lower level. Also, the largest relative improvement of the proposed method compared to the baseline implementation can be seen for the two lane change action classes.

This may be explained by the dataset statistics together with shape of lane change trajectories. While lane changes as well as turns are underrepresented in the training data, turns have a much more distinct trajectory shape and thus

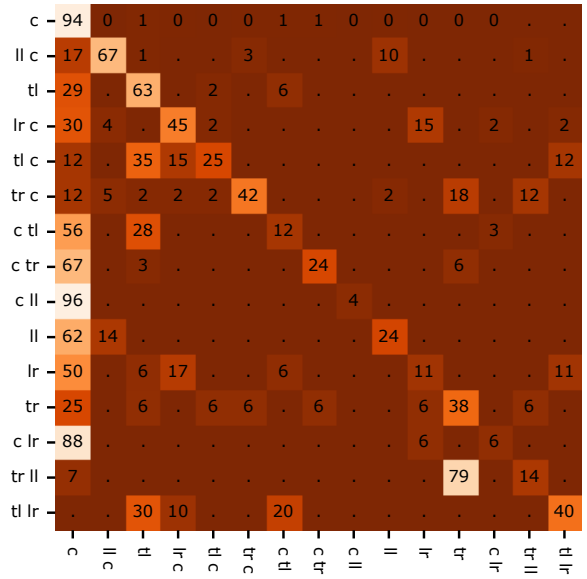

Fig. 4: Confusion matrix for top-1 ordered sequence predictions with the x-axis showing predicted classes and y-axis showing groundtruth classes. All numbers in %, normalized that rows sum to 100.

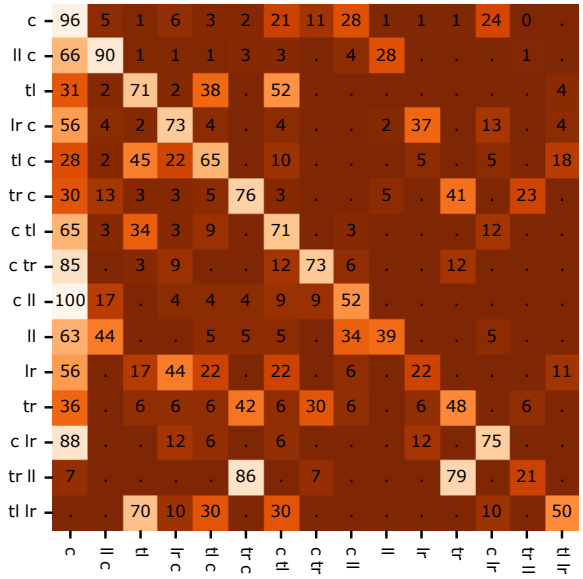

Fig. 5: Confusion matrix for top-2 ordered sequence predictions with the x-axis showing predicted classes and y-axis showing groundtruth classes. All numbers in %, normalized that rows sum to 200.

can be detected more easily, even without modeling them explicitly.

B. *N*-Most Likely Ordered Sequences

Results of evaluating our proposed as well as the best performing baseline method on the manually annotated ordered sequences are provided in Table III. In contrast to the baseline model, where the cruise action class is overrepresented, our proposed method predicts a more diverse set of action sequences. While a relatively high top1-accuracy is observed for sequences that require detecting a maneuver, e.g. (tl), (ll, c), (lr, c), sequences that involve the prediction of maneuvers such as (c, ll) or (c, lr) show a high improvement in the top2-accuracy. The observation that some action classes heavily improve in top2-accuracy is in line with the commonly accepted assumption that agent prediction needs to be multimodal to account for the uncertainty of the future.

Fig. 4 and 5 show the confusion matrix for the top-1 and top-2 sequence prediction, with sequences ordered according to their frequency in the dataset. For top-1 prediction, the classification errors mostly stem from assigning sequences with an incorrect second action, e.g. (tl) instead of (tl, c), which confirms that prediction is a much harder task than classification. Still, the top-2 predictions allow for correcting for many of these errors, resulting in substantially higher values on the diagonal.

C. Ablation Study

An ablation study is conducted to evaluate the impact of the separate modules. Ablated methods are compared using the mean AP on the validation dataset, with results shown in Table IV. To investigate the influence of input representation the input tensor is grouped into following modules:

Input representation				
Target	Social	Map	Augmentation	mean AP
✓			✓	0.375
✓	✓		✓	0.490
✓		✓	✓	0.602
✓	✓	✓		0.590
✓	✓	✓	✓	**0.614**

TABLE IV: Ablation study on the action prediction network.

The target agent information (Target) includes positions and velocities of the target agent $(\mathbf{1}_{target}, \mathbf{v}_{target})$. Social context (Social) comprises of the positions and velocities of the other agents $(\mathbf{1}_{other}, \mathbf{v}_{other})$. Finally, map information (Map) contains the layer representing the lane centerlines $\mathbf{m}$.

Besides ablating the input representation, we show that traffic agent action prediction can profit from the presented data augmentation approach of rotating the complete input representation by a small random angle.

The results confirm the usefulness of providing all three sources of information jointly, with the map having the biggest impact. Interestingly, adding social information to a representation that does not contain a map, has substantially higher impact than adding it to a representation that does contain a map. This indicates that there is redundant information available in the map and social context the network is able to use.

D. Dataset Scale study

The second ablation study investigates the relevance of the dataset size for the action prediction task while using the full model. Thus, the network is trained with three subsets of our data that reflect 50%, 10% and 5% of the full set of action

Dataset proportion	mean AP
100%	**0.614**
50%	0.571
10%	0.522
5%	0.493

TABLE V: Ablation study on the action prediction dataset.

sequences. The dataset reduction is implemented by skipping the corresponding number of samples, to avoid changing the dataset statistics. Epoch length and sample weights are kept constant, such that the training parameters are stable. The results of the ablation study, shown in Table V, show a large difference in performance between the different scales, indicating that the amount of data has major impact on the prediction performance. Furthermore, the considerable improvement from 50% of the data to the full dataset allows for the conclusion that the performance did not saturate at the scale present in the Argoverse dataset and larger amounts of data could benefit the community.

VII. Conclusion and Future Work

In this paper we investigated the task of predicting high-level actions of vehicles in urban environments. To make this task viable, we proposed an algorithm to automatically extract action sequences using HD-maps, from the public Argoverse [1] dataset. Furthermore, we proposed an action prediction network, that predicts the future actions sequence considering agent, map, and social information. The network is completely based in rendered images and can be trained in an end-to-end fashion using the automatically generated large scale action sequence dataset. We showed that our action prediction model, together with our dataset, can significantly outperform existing methods that are adapted from trajectory prediction. This additionally shows that action prediction is not solved by trajectory prediction. Thus, in our future work we will investigate how action prediction and trajectory prediction can be combined to get the best of both worlds.

Acknowledgement: The work is supported by Toyota Motor Europe via the research project TRACE-Zürich.

References

[1] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3D tracking and forecasting with rich maps," in *Conference on computer vision and pattern recognition (CVPR)*, 2019.

[2] B. Morris, A. Doshi, and M. Trivedi, "Lane change intent prediction for driver assistance: On-road design and evaluation," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[3] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena, "Car that Knows Before You Do: Anticipating Maneuvers via Learning Temporal Driving Models," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[4] S. Hecker, D. Dai, and L. Van Gool, "End-to-End Learning of Driving Models with Surround-View Cameras and Route Planners," in *European Conference on Computer Vision (ECCV)*, 2018.

[5] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K.-D. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," in *IEEE Intelligent Vehicles Symposium (IV)*, 2014.

[6] A. Khosroshahi, E. Ohn-Bar, and M. M. Trivedi, "Surround vehicles trajectory analysis with recurrent neural networks," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[7] C. Laugier, I. E. Paromtchik, M. Perrollaz, M. Yong, J.-D. Yoder, C. Tay, K. Mekhnacha, and A. Nègre, "Probabilistic Analysis of Dynamic Scenes and Collision Risks Assessment to Improve Driving Safety," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 4, pp. 4–19, 2011.

[8] M. Schreier, V. Willert, and J. Adamy, "Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems," in *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014.

[9] A. Houenou, P. Bonnifait, V. Cherfaoui, and Wen Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[10] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? A Unified Framework for Maneuver Classification and Motion Prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 129–140, June 2018.

[11] N. Deo and M. M. Trivedi, "Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[12] S. Casas, W. Luo, and R. Urtasun, "IntentNet: Learning to predict intention from raw sensor data," in *conference on robot learning, CoRL 2018*, ser. Proceedings of machine learning research, 2018.

[13] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 2165–2174.

[14] J. Hong, B. Sapp, and J. Philbin, "Rules of the road: predicting driving behavior with a convolutional model of semantic interactions," in *The IEEE conference on computer vision and pattern recognition (CVPR)*, 2019.

[15] X. Li, X. Ying, and M. C. Chuah, "GRIP: Graph-based Interaction-aware Trajectory Prediction," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3960–3966.

[16] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst," *arXiv:1812.03079 [cs]*, Dec. 2018.

[17] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 261–268.

[18] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 961–971.

[19] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 2255–2264. [Online]. Available: https://ieeexplore.ieee.org/document/8578338/

[20] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "SoPhie: an attentive GAN for predicting paths compliant to social and physical constraints," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2019.

[21] B. D. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J.-W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in *IEEE 20th international conference on intelligent transportation systems (ITSC)*, 2018.

[22] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Sept. 2013.

[23] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Lyft level 5 AV dataset 2019," 2019, https://level5.lyft.com/dataset/.

[24] 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," *arXiv preprint arXiv:1903.11027*, 2019.