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

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-7281-6212-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/IROS45743.2020.9341327

Multiple Trajectory Prediction with Deep Temporal and Spatial Convolutional Neural Networks

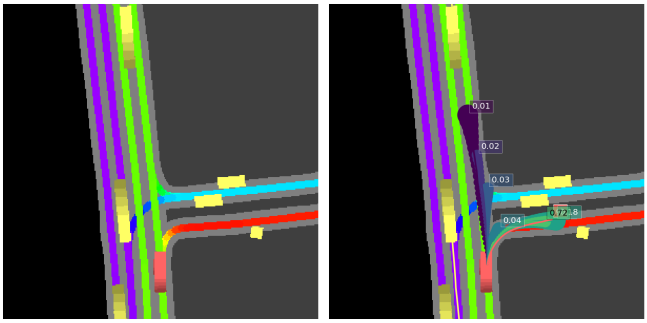
Jan Strohbeck[1], Vasileios Belagiannis[1], Johannes Müller[1],
Marcel Schreiber[1], Martin Herrmann[1], Daniel Wolf[1] and Michael Buchholz[1]

***Abstract*— Automated vehicles need to not only perceive their environment, but also predict the possible future behavior of all detected traffic participants in order to safely navigate in complex scenarios and avoid critical situations, ranging from merging on highways to crossing urban intersections. Due to the availability of datasets with large numbers of recorded trajectories of traffic participants, deep learning based approaches can be used to model the behavior of road users. This paper proposes a convolutional network that operates on rasterized actor-centric images which encode the static and dynamic actor-environment. We predict multiple possible future trajectories for each traffic actor, which include position, velocity, acceleration, orientation, yaw rate and position uncertainty estimates. To make better use of the past movement of the actor, we propose to employ temporal convolutional networks (TCNs) and rely on uncertainties estimated from the previous object tracking stage. We evaluate our approach on the public "Argoverse Motion Forecasting" dataset, on which it won the first prize at the Argoverse Motion Forecasting Challenge, as presented on the NeurIPS 2019 workshop on "Machine Learning for Autonomous Driving".**

I. Introduction

Predicting the future motion of other traffic participants, implicitly or explicitly as in this work, is key to behavior planning in automated vehicles [1], [2], [3], [4]. This holds in particular for challenging maneuvers like lane merging on highways or crossing an intersection, where accounting for the behavior of other road users is vital for the safe and successful execution of the planned manoeuvre. Furthermore, an adequate long-term prediction of other traffic participants enables behavior planning on an extended planning horizon, so that safety is increased as critical situations can be detected early and reacted upon, and passenger comfort as well as driving efficiency can be increased.

In literature, motion prediction of road users is a well-studied field [5] dating back to early approaches using a Kalman filter in combination with kinematic models [6]. These classical approaches, however, cannot adequately account for driver behavior and the driving environment and thus perform badly for long-term predictions. Extensions of these classical approaches, e.g. with Gaussian processes [7], solve some of these problems, but come with other limitations, as highlighted in [5]. Recently, advances in neural networks [8] as well as the availability of motion forecasting datasets, e.g. [9], gave rise to approaches based on neural networks [10], [11], [12], [13].



(a) Synthetic input image. (b) The input image with the predicted trajectories overlaid.

Fig. 1: Examples of the input and output of the prediction network. The input image (a) encodes other traffic participants and their previous motion (yellow), lane geometry (colored centerlines and light grey outline) and drivable area (darker grey). In (b), six forecasted trajectory hypotheses are displayed along with their estimated probabilities and estimated position uncertainties. The circles are centered on the predicted positions and the radius corresponds to two times the predicted standard deviation. The thin red line shows the ground truth trajectory.

In this work, we build upon the approach presented by Cui et al. [13] that creates synthetically generated actor-centric images and the actor state as the input for a mainly convolutional neural network. An example of such synthetic images can be seen in Fig. 1a. An appealing key idea of [13] is the introduction of a novel loss function, which allows the network to learn predicting multiple hypotheses efficiently without additional annotation for manoeuvre classes. Motivated by the good results achieved by Cui et al., we enhance and further improve the performance of that approach with a number of new innovations.

Firstly, we extend the network to allow producing estimates for the uncertainty inherent in the predicted trajectories. This can be seen in Fig. 1b, where the uncertainty is visualized as circles around the predicted positions. Secondly, we add estimated uncertainties from a previous tracking stage to the network input to further support the assessment of the inputs; and thereby improve the quality

*Part of this work has been conducted as part of ICT4CART project which has received funding from the European Union's Horizon 2020 research & innovation program under grant agreement No. 768953. Content reflects only the authors' view and European Commission is not responsible for any use that may be made of the information it contains.

[1]The authors are with the Institute of Measurement, Control and Microtechnology, Ulm University, D-89081 Ulm, Germany
Email: <firstname>.<lastname>@uni-ulm.de, except for johannes-christian.mueller@uni-ulm.de

of the predictions, as well as the quality of the hypothesis probabilities and the estimated uncertainties. Thirdly, we introduce multi-task learning with additional loss functions to cover additional vehicle state dimensions, namely velocities, accelerations, orientation and yaw rate. Lastly, we include the past trajectory of the actor as an input to the network and process it with a temporal convolutional network (TCN). As a result, the proposed network architecture explicitly considers the past trajectories.

We evaluate our approach on the publicly available "Argoverse Motion Forecasting" dataset [9]. As announced at the NeurIPS 2019 workshop on "Machine Learning for Autonomous Driving", our approach won the first place on the public Argoverse Motion Forecasting challenge in 2019.

II. Related Work

Forecasting future motion of traffic participants has been addressed extensively in the past, and is currently a very active research topic. For short prediction horizons of one second or less, simple kinematic models are often used, e.g. in Kalman filters [6], as the short-term motion of vehicles is mostly determined by kinematics. However, for larger prediction horizons of several seconds, vehicle motion is also largely dependent on other traffic participants, driver intentions and road geometry. In order to simplify the prediction, several approaches attempt to explicitly model the possible maneuvers a vehicle can perform [12], [14]. In contrast, the approach presented in this work does not explicitly model possible maneuvers, but relies on a neural network to learn all possible maneuvers from training data. Additionally, Ziegler et al. [15] propose to generate a list of possible future paths that a vehicle can take using constraints from map data. The approach in this work also requires map data, but it is only used as an input to the neural network, and not to explicitly generate trajectory proposals. We thereby do not constrain the trajectory hypotheses to always follow driving lanes or maintaining a certain distance from the lane boundaries, as many human drivers also do not obey these constraints in the real world. Many other approaches only predict a single trajectory per traffic participant [16], [14], while our approach generates multiple possible future trajectories for each actor and scores them with probabilities.

The approaches in [11] and [17] propose the prediction of the vehicle environment using dynamic occupancy grid maps. While these approaches are also able to generate multiple future paths for each actor, the output representation is an occupancy grid map, where the information of the single actors is lost. The approach presented in this work, however, generates multiple trajectories as explicit lists of future states, which include position, velocities, acceleration, etc.

Motion forecasting methods are often based on recurrent neural networks (RNNs), because of their capability to process and generate sequences, which are in these applications a sequence of objects states, i.e. trajectories. Mercat et al. [10] use a long short-term memory (LSTM) [18] to predict multiple future trajectories per traffic actor, jointly for all traffic actors in a scene, and scoring the trajectories using self-attention layers. The approach in [19] also utilizes LSTMs and attention for jointly predicting trajectories, but introduces latent variables for generating multiple trajectories per traffic actor. In our work, we employ a simpler, feed-forward architecture that also predicts multiple trajectories for each actor and can be parallelized across all traffic actors.

Cui et al. in [13] encode map information and the state of the traffic participants in synthetically generated actor-centric images and use it as the input to a mainly convolutional neural network. In addition, they define a novel Multiple-Trajectory Prediction (MTP) loss, which allows the network to learn multiple hypotheses efficiently without additional annotation for manoeuvre classes. As shown in their work, predicting multiple possible trajectories for each traffic participant leads to better results than generating just one trajectory. This is due to the problem of mode averaging, which means that multiple possible maneuvers, e.g. turning left or going straight, are averaged into a single mode, which is in some cases (e.g., at intersections) an unrealistic manoeuvre.

III. Method

In this section, we outline the problem of motion prediction for traffic participants, shortly recap on the method proposed by Cui et al. in [13] and then present our modifications to it.

A. Problem Formulation

In order to train a neural network for trajectory prediction, we need a set of recorded trajectories of traffic participants. These can be extracted from data recorded from a vehicle equipped with sensors such as cameras, lidars and radars, infrastructure sensors or drones. Systems to detect objects in sensor data and track them in time are a necessity for today's autonomous driving [20], so we assume such a system is already in place. Many such systems employ a variant of the Kalman Filter, e.g. an LMB filter [21] [22]. With such a system in place, trajectories of traffic participants can then be easily recorded and used for training a prediction system [9], as it is shown in this work.

We denote the state of an individual traffic participant i at the time step j with $\mathbf{S}_{ij}$. The states comprise all data that is inferred by the tracking system, i.e. position in x and y coordinates, velocity v, acceleration a, orientation ϕ and yaw rate $\dot{\phi}$. The task is then to predict the future states $[\mathbf{S}_{i(j+1)}, \ldots, \mathbf{S}_{i(j+H)}]$ for each actor i, up to the prediction horizon H, given the current and past states of all traffic participants and map data $\mathcal{M}$. The prediction horizon H is typically equivalent to several seconds, as this is required for the motion planning to handle complex urban traffic scenarios. $\mathcal{M}$ is a high-definition map, containing road polygons, lane centerlines and polygons indicating drivable area, to give additional context information to the neural network. Such maps are already employed in most state-of-the-art motion and behavior planning systems in automated vehicles. The maximum number of past states that we process in this approach is denoted by T. Note that there may not be exactly

T past states available for every traffic participant, as traffic actors may appear in and disappear from the perceptible field of the sensors. We require past data to be equally-spaced in time, e.g. at a constant sampling frequency of 10 Hz.

B. MTP Background

To aid the understanding of our approach, we recap on the MTP method proposed by Cui et al. that composes the basis of this work. There, rasterized images, which encode information about the scene in a bird's eye view, are used as the main input to the prediction network [13]. Since each traffic actor is predicted individually, one input image is generated for each actor, using the information of a high-definition (HD) map and the other traffic participants. From the HD map, drivable area, road polygons and lane centerlines are rasterized on top of each other. While drivable area and road polygons are rendered with separate fixed colors, the color of the lane centerlines encodes the direction of the lane, relative to the actor of interest. The actor of interest is always rendered centered near the bottom of the image, pointing upward, and the other image contents are rotated and shifted accordingly. The current state $\mathbf{S}_{ij}$ and future states are also adjusted according to this transformation. Other traffic participants are also rendered, but in a different color (yellow) than the actor of interest (red). Also, all traffic participants' positions in previous time steps are rendered with decreasing brightness to produce a "fading" effect. The input image (Fig. 1a) is then processed by a CNN in order to generate features.

Besides the image, the network in [13] also receives the current state of the actor of interest at time t_j. In [13], this current state comprises the current (observed) velocity, acceleration and yaw rate of the actor. These features are concatenated with the features generated by the CNN and then fed into a series of two linear layers, which generate the output vector. This output vector can be reshaped to be a set of M possible trajectory hypotheses $\tilde{\tau}_{imj}$, where $m = 1 \dots M$ for the actor of interest i, as well as an estimated probability for each hypothesis to actually occur. M is a design parameter and fixed after the network creation. The output of the network is a vector, which can be reshaped to be of dimensions $M \times (2H+1)$, which can be interpreted as M trajectories with predictions over H time steps, where each prediction contains x and y coordinates, plus an additional probability per trajectory.

The loss is backpropagated only for the best matching trajectory and the predicted probabilities:

$$\mathcal{L}_{ij}^{MTP} = \alpha_{class}\mathcal{L}_{ij}^{class} + \sum_{m=1}^{M} I_{m=m^*}\, L(\tau_{ij}, \tilde{\tau}_{imj}), \quad (1)$$

where $\mathcal{L}_{ij}^{class}$ is a cross-entropy loss that is calculated for the estimated probabilities of the hypotheses and α_{class} is a tunable hyperparameter. $I_{m=m^*}$ is a binary function, which evaluates to 1 if $m = m^*$ and to 0 otherwise. The mode m^* is the mode that fits best to the ground truth trajectory:

$$m^* = \underset{m \in \{1,\dots,M\}}{\operatorname{argmin}} \; dist(\tau_{ij}, \tilde{\tau}_{imj}). \quad (2)$$

$dist(\tau_{ij}, \tilde{\tau}_{imj})$ can be any single-mode distance function, e.g. an MSE between the predicted trajectory positions and the ground truth positions. In [13], this can also be a function comparing the angle formed between the end positions of both trajectories and the actor position at time j. $L(\tau_{ij}, \tilde{\tau}_{imj})$ is a single-trajectory loss between the trajectory of mode m and the ground truth. In [13], this is an MSE loss between the x and y positions of both trajectories.

C. Our Approach

Similar to [13], we use rasterized images as the main input to the network, as this is an efficient method to represent map information and the dynamic surrounding of the actor of interest. We also use MobileNetV2 [23] as the CNN, but any other CNN could also be used instead. The overall structure of our network is shown in Fig. 2. Besides the image, we also use current and past state of the actor of interest as inputs to the network. A TCN extracts features from the past and current states, which are then concatenated with the features from the CNN and the current state. A series of three linear layers calculates the final output, which are the M predicted trajectories.

In the following, we describe the modifications we made compared to the approach in [13].

1) Using inherent uncertainties: In [13], the actor state that is fed into the network in addition to the rasterized image encodes the current velocity, acceleration, and yaw rate. We propose to additionally make use of the uncertainties that are estimated in Kalman filter based object tracking systems. More specifically, we use the variances $\mathcal{P}$ that are estimated for all components of the actor state. This means that the dimension of the state input and the input of the TCN is increased to be $2S$. Adding the variances as an additional input can help the network to better estimate if trajectories are at all possible due to uncertainty in the state, and to estimate the resulting probabilities and uncertainties, thereby improving the overall quality of the generated trajectories.

2) Utilizing past trajectories explicitly: Furthermore, we propose to use not only the current state of the actor, but to also explicitly use states from previous time steps for the prediction. This is motivated by the assumption that the current state does not fully encode the history of the actor of interest, and that more information can be gained for the prediction of the actor by processing its past states. While the network in [13] receives the motion history implicitly via the "fading" effect in the rasterized images, an explicit usage of the data as an additional input can be superior. The additional data contains not only past positions, but also the past velocities, accelerations, orientation, yaw rate and tracking uncertainties, which are difficult to encode in and extract from rasterized images.

For processing this information, we propose to use a temporal convolutional network (TCN) [24]. TCNs are, similar to recurrent neural networks (RNNs), capable to process sequences and extract temporal features from them. TCNs showed in many tasks superior performance compared to RNNs as shown in [25], so we decided to use them

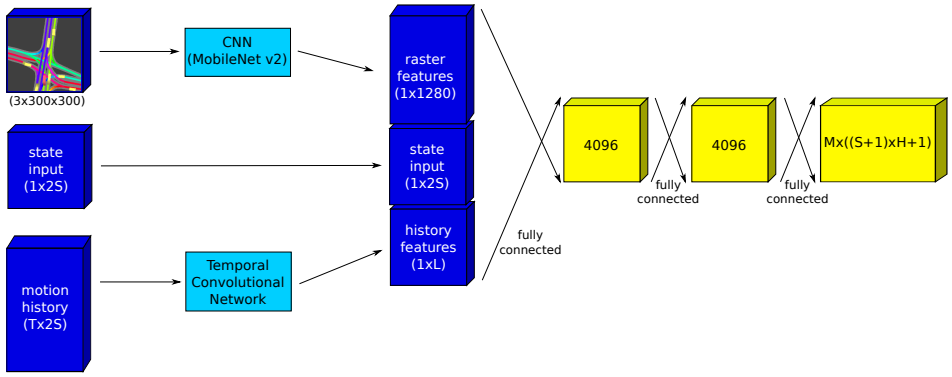


Fig. 2: Structure of the proposed network. The inputs are a rasterized image of the static and dynamic environment of the actor to predict, the current state of the actor with uncertainties estimated by a previous object tracking stage, and the past states of the actor, also with uncertainties. The image is processed by a CNN to produce features, which are concatenated with the current state and features extracted from the past states by a TCN. A series of linear layers produces the predicted trajectories, comprised of future states and position uncertainties.

in our approach. The TCN in our network receives the past and current states $[\mathbf{S}_{i(j-T_{ij}+1)}, \dots, \mathbf{S}_{ij}]$, as well as the corresponding state uncertainties, as the input. T_{ij} is the number of available past states for this actor, which can range from one to the maximum T. The TCN generates a number L of output features, describing information extracted from the past trajectory. The TCN features are later combined with the features obtained from the CNN and with the current state of the actor of interest, to then predict the output trajectories.

3) Multi-task learning: The model output are a set of M possible hypotheses for the actor of interest and estimated probability for each hypothesis to actually occur, same as in [13]. In contrast to [13], we propose to generate not only future position estimates, but also to predict all the other features of the states $[\mathbf{S}_{i(j+1)}, \dots, \mathbf{S}_{i,(j+H)}]$, i.e. velocities, accelerations, etc. This multi-task learning can help the network to learn faster and generalize better, as shown in [26], [27]. The resulting single-mode loss for Eq. (1) is then

$$L(\tau_{ij}, \tilde{\tau}_{imj}) = \alpha_{pos} L^{pos}_{imj} + \alpha_v L^{v}_{imj} + \alpha_a L^{a}_{imj} + \alpha_\phi L^{\phi}_{imj} + \alpha_{\dot{\phi}} L^{\dot{\phi}}_{imj}. \quad (3)$$

We employ MSE losses for all of these loss terms L_{imj}, except for L^{pos}_{imj}, which is covered in the next section. All α values are tunable hyperparameters. Note that we actually use two values $[\cos(\phi), \sin(\phi)]$ to represent the orientation of the vehicle, which proved to be easier for the neural network to estimate than the raw euler angle ϕ, as euler angles are prone to ambiguities. As multiple hypotheses are generated, we use the MTP approach to only calculate gradients for the trajectory that is closest to the ground truth, as stated in Eq. (1).

4) Estimating position uncertainties: Predicting uncertainties for each trajectory point can be helpful for the following motion planning stage, as it is another indicator of how much the predicted trajectory can be trusted, aside from its predicted probability. In [13], an MSE loss was employed for L^{pos}_{imj} and no uncertainties were estimated. In [16], an alternative loss function which learns the position as well as the uncertainty of the prediction simultaneously was presented, which bases on the assumption that the predicted position errors are sampled from a half-normal distribution. There, it is used to predict only a single trajectory, but it can easily be applied to multiple trajectory prediction:

$$L^{pos}_{imj} = \sum_{h=1}^{H} \left(\frac{d^2_{im(j+h)}}{2\hat{\sigma}^2_{im(j+h)}} + \log \hat{\sigma}_{im(j+h)} \right). \quad (4)$$

Here, $\hat{\sigma}$ denotes the uncertainties (standard deviations) estimated by the network, and d are the actual euclidean position distances between the ground truth trajectory and the estimated trajectory at each predicted point $j+h$ of the generated trajectory m. Generating the uncertainty estimates is implemented by increasing the size of the generated output vector per predicted trajectory, so that it has dimensions $M \times ((S+1) \cdot H + 1)$.

As an alternate method for predicting uncertainties, we also explored the possibility to use an MSE loss for the position (as in [13]), and employing an additional MSE loss for the uncertainty:

$$L^{pos}_{imj} = \frac{1}{H} \sum_{h=1}^{H} \left(\left|\left| \mathbf{x}_{i(j+h)} - \hat{\mathbf{x}}_{im(j+h)} \right|\right|_2 + \alpha_\sigma \left(d_{im(j+h)} - \hat{d}_{im(j+h)} \right)^2 \right). \quad (5)$$

Here, $\mathbf{x}_{i(j+h)}$ is the vector of x and y positions of the actor i at timestep $j+h$, while $\hat{\mathbf{x}}_{im(j+h)}$ are the corresponding predicted position values. α_σ is a hyperparameter that controls the influence of the uncertainties on the total loss. $\hat{d}_{im(j+h)}$ is calculated from $\hat{\sigma}_{im(j+h)}$ and should ideally be equal to $\mathbb{E}\left[d_{im(j+h)}\right]$, which, using the half-normal assumption, is equal to $\frac{\sigma\sqrt{2}}{\sqrt{\pi}}$. This means that $\hat{d}_{im(j+h)} = \frac{\hat{\sigma}_{im(j+h)}\sqrt{2}}{\sqrt{\pi}}$ can be used to calculate $\hat{d}_{im(j+h)}$ for Eq. (5). This approach is closer to the implementation in [13], as it also uses MSE

loss for position.

We report results for both the modified MSE position loss and the combined trajectory and uncertainty loss in the following section, where we refer to the variant using the position loss of Eq. (4) as Variant A, and to the variant using the position loss of Eq. 5 as Variant B.

IV. Experiments

Here, we shortly describe our experimental setup, outline the metrics that we use for the evaluation of our method, and then highlight our results on the Argoverse Motion Forecasting dataset. We also provide details about the actual implementation.

A. Experimental Setup

We evaluate our approach on the Argoverse Motion Forecasting Dataset v1.1 [9]. This dataset consists of 323,557 challenging scenarios of 5 seconds each, sampled at $10\,\mathrm{Hz}$, which were extracted from more than 1000 hours of driving on public roads in Miami and Pittsburgh. In each scenario, the positions of all detected traffic actors are given, and the movement of one traffic actor of interest shall be predicted. Aside from the large number of scenarios, an advantage of this dataset is the availability of detailed HD maps, which are required for implementing our approach.

The dataset is split into 205,942 scenarios for training, 39,472 scenarios for validation and 78,143 scenarios for testing. In the test dataset, labels are not available, but a dedicated server can be used to upload results and obtain scores of the proposed approach. This server was also used for the Argoverse Motion Forecating challenge, which was a competition held by Argo AI from September 2019 to December 2019 [28]. In the competition, two seconds of recorded trajectories were available per scenario as past states of the traffic participants, while three seconds of future trajectories should be predicted for the actor of interest in each scenario of the test dataset. In our experiments, we therefore use the same split between past states and future states, that is two seconds of history and three seconds as the label/ground truth. Note that our approach can also be applied for larger values of the prediction horizon H and is not restricted to three seconds.

As the trajectories in the data are noisy and contain only positions, we run an extended Kalman filter (EKF) with a Constant-Turn-Rate-and-Acceleration (CTRA) state model to estimate the missing quantities and smooth the recorded trajectories. Since the dataset does not contain object extents or classifications, the generation of the synthetic images had to be implemented using hypothesized object extents. Hence, the dataset is not optimal for the implementation of this approach, but as shown in the following sections, it nonetheless performs well, and might perform even better on datasets which contain these additional features. In our experiments, we noticed that a large part of the trajectories in the dataset are going more or less straight, which lead to worse performance of the network in predicting turning maneuvers. To counteract this, we added random rotations to the images, so that the extraction of the map features is more robust.

B. Evaluation Metrics

As evaluation metrics, we use the metrics described in [9], which are also used in the Argoverse challenge. The minimum Final Displacement Error (minFDE) calculates the final displacement (at time t_H) of the best of the top-K most probable hypotheses:

$$\text{minFDE}_K = \min_{k \in \{1 \dots K\}} \sqrt{(x_H - \hat{x}_{k,H})^2 + (y_H - \hat{y}_{k,H})^2}.$$

This means that with varying K, the number of hypotheses that are evaluated can be modified. The minimum Average Displacement Error (minADE) then calculates the average displacement over the whole prediction of the hypothesis with minimum FDE. The Drivable Area Compliance (DAC) calculates the average percentage of the top-K hypotheses that are on the drivable area at all times. Miss Rate (MR) calculates the percentage of scenarios in which none of the top-K hypotheses is closer than a threshold θ_M to the ground truth trajectory. For the Argoverse challenge, the threshold θ_M was set to 2.0 meters.

The metrics minADE and minFDE encourage the generation of multiple trajectories in the network, as the network then has multiple attempts at generating a trajectory that is near to the ground truth. As shown by [13], using a metric that just compares the most probable hypothesis to the ground truth favors single-hypothesis models, which optimize for averaged prediction error while generating unrealistic trajectories in many cases.

C. Implementation Details

The network was implemented in PyTorch [29]. We used the Rectified Adam Optimizer [30] with Lookahead [31], and stopped training after 40 epochs. The initial learning rate was set to 10^{-3} and reduced by a factor of 10 every 12 epochs. For the challenge, we trained on both training and evaluation set to further improve results, but the results in this paper are achieved by using only the training dataset. The values for the hyperparameters used for both variants are listed in Table II. We use the Mish activation function [32] throughout the network (except for the pre-trained MobileNetV2 CNN). The uncertainties are passed through a softplus activation, in order to ensure that they are always positive. For the TCN, we employ a similar structure to the one defined in [25], with 5 layers of convolutional blocks with a kernel size of 2, which reduces the number of features from $2S = 14$ to 6, 6, 4, 4 and finally $L = 2$ features. Using 5 layers ensures that the receptive field of the TCN is large enough to cover enough past states, as the receptive field increases exponentially with the depth of the network.

The model runs inference in $2.0\,\mathrm{ms}$ with batch size $B = 1$ and in $7.4\,\mathrm{ms}$ with $B = 32$, when using NVIDIA TensorRT acceleration (generation of images and uploading/downloading data to/from GPU excluded) with ONNX Runtime on an NVIDIA Geforce 1080 Ti.

TABLE I: Results of our approach (Variant A and B) on the test dataset, compared to the Argoverse Baseline from [9] and our re-implementation of the MTP model from [13].

Model	$K=1$				$K=3$				$K=6$			
	minADE_1	minFDE_1	DAC_1	MR_1	minADE_3	minFDE_3	DAC_3	MR_3	minADE_6	minFDE_6	DAC_6	MR_6
Argoverse Baseline	2.96	6.81	0.90	0.81	2.67	6.17	0.90	0.75	2.34	5.44	0.90	0.69
MTP ($M=3$)	2.11	4.63	**0.98**	0.68	1.28	2.41	**0.98**	0.46	1.28	2.41	**0.98**	0.46
MTP ($M=6$)	2.14	4.68	**0.98**	0.68	1.35	2.59	**0.98**	0.37	1.04	1.70	0.97	0.26
Variant A (Eq. (4))	**1.90**	**4.19**	**0.98**	**0.63**	**1.19**	**2.28**	**0.98**	**0.31**	**0.94**	**1.54**	0.97	**0.21**
Variant B (Eq. (5))	1.99	4.37	**0.98**	0.66	1.22	2.34	**0.98**	0.33	0.97	1.60	0.97	0.24

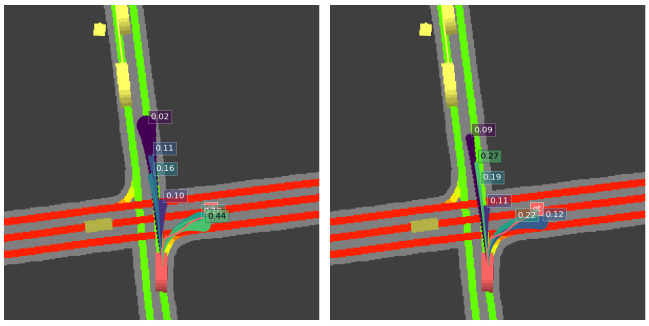
TABLE II: Values for the hyperparameters α from Eq. (1) and Eq. (3).

Model	α_{class}	α_{pos}	α_v	α_a	α_ϕ	$\alpha_{\dot{\phi}}$	α_σ
Variant A (Eq. (4))	0.2	1.0	0.05	0.2	1.0	1.0	–
Variant B (Eq. (5))	0.4	1.0	0.1	0.3	1.0	1.0	0.2

D. Results

In Table I, we report the results obtained by our approach, compared to the Argoverse Baseline from [9] and to our re-implementation of the MTP model from [13]. For the MTP model, we use two configurations, one with $M=3$ and one with $M=6$. With Variant A, we refer to our model using the position loss in Eq. (4), while Variant B uses the position loss in Eq. (5).

We report all metrics for $K=1,3,6$. Furthermore, our evaluations show that with our training setup, the MTP model achieves heavily improved results, compared to the Argoverse Baseline. In addition, increasing M from 3 to 6 significantly improves all metrics at $K=6$, as the metrics reward generating more trajectory hypotheses. However, our approaches with variants A and B achieve even better results in most metrics, outlining the benefits of the new network structure and our training setup. Variant A performs slightly better than Variant B in most metrics. In spite of that, for assessing the quality of the predictions, the predicted uncertainty must be evaluated as well.



(a) Variant A (b) Variant B

Fig. 3: Visualization of the predicted trajectories with estimated position uncertainties. The circles are centered on the predicted positions and the radius corresponds to two times the predicted standard deviation.

In order to visualize the predicted uncertainties for both variants, we look at an exemplary intersection scenario from the validation set, as depicted in Fig. 3. It is shown that while both variants produce similar predictions, the estimated uncertainties of the predictions are quite different. For evaluating the estimated trajectory uncertainty quantitatively, we use reliability diagrams as in [16]. These are generated by measuring the observed distance between the predicted points of the best matching trajectory versus the ground truth, and then computing the fraction of observed errors that fall inside the expected range indicated by the predicted variance. This means that for a predicted variance of 1, we would expect about 68% of observed errors to be within the predicted one sigma, as we assume a half-normal distribution of observed errors. A reliability diagram for the variants A and B is depicted in Fig. 4, where the observed error fraction is plotted versus the predicted error fraction. The closer the plot is to a diagonal line, the better the estimated variances resemble the true error distribution. From the graph, we can see that both variants are quite well aligned with the reference line, while Variant B produces slightly better uncertainty estimates than A. Variant A, however, shows better performance with its predictions according to Table I, while the quality of the predicted uncertainties is equally good as Variant B. The uncertainties of both variants are reliable enough to be used in a subsequent motion planning stage. To further improve the quality of the uncertainties, recalibration techniques such as in [33] can be employed.

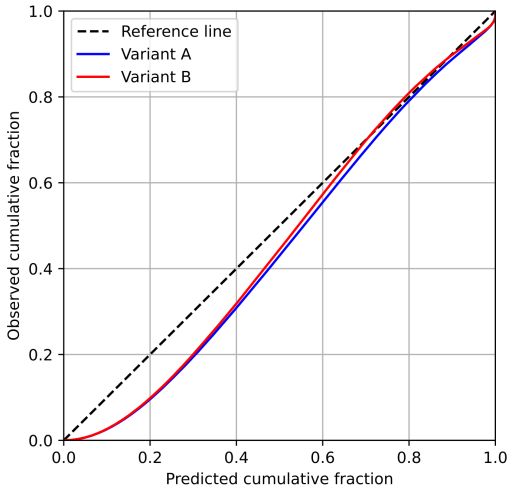


Fig. 4: Reliability diagram for both variants at $K=6$. Both variants are quite well aligned with the ideal reference line.

V. Conclusions

In this work, we presented an approach for predicting the future trajectories of traffic participants, which makes use of uncertainties that are inherent in the process of detecting and tracking objects. We also exploit their previous motion history via a temporal convolutional network, and employ multi-task learning to further improve the results. Finally, we have evaluated two variants of our method for learning the uncertainty of the prediction. Our approach delivers state-of-the-art results as it won the first place at the Argoverse Motion Forecasting Challenge, as presented on the NeurIPS 2019 workshop on "Machine Learning for Autonomous Driving". In our future work, we want to explore multiple trajectory prediction for different domains such as human motion prediction, and extend existing approaches such as [34]. Also, we want to evaluate the potential of robust regression methods as proposed in [35] for the prediction.

References

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, March 2016.

[2] J. Schulz, C. Hubmann, J. Löchner, and D. Burschka, "Interaction-aware probabilistic behavior prediction in urban environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 3999–4006.

[3] J. Müller and M. Buchholz, "A risk and comfort optimizing motion planning scheme for merging scenarios," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct 2019, pp. 3155–3161.

[4] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," 2018.

[5] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH journal*, vol. 1, no. 1, p. 1, 2014.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[7] J. Wiest, *Statistical long-term motion prediction*. Universität Ulm, Institut für Mess-, Regel-und Mikrotechnik, 2016.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] M. Chang, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 8740–8749.

[10] J. Mercat, T. Gilles, N. E. Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Multi-head attention for multi-modal joint vehicle motion forecasting," *arXiv preprint arXiv:1910.03650*, 2019.

[11] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2056–2063.

[12] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/IVS.2018.8500493

[13] H. Cui, V. Radosavljevic, F. Chou, T. Lin, T. Nguyen, T. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 2090–2096.

[14] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, "Vehicle trajectory prediction by integrating physics- and maneuver-based approaches using interactive multiple models," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5999–6008, July 2018.

[15] J. Ziegler, *et al.*, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, Summer 2014.

[16] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, and J. Schneider, "Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks," *arXiv preprint arXiv:1808.05819*, 2018.

[17] M. Schreiber, S. Hoermann, and K. Dietmayer, "Long-term occupancy grid prediction using recurrent neural networks," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 9299–9305.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[19] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 15 424–15 434. [Online]. Available: http://papers.nips.cc/paper/9676-multiple-futures-prediction.pdf

[20] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–20, 2020.

[21] S. Reuter, B. Vo, B. Vo, and K. Dietmayer, "The labeled multi-bernoulli filter," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3246–3260, June 2014.

[22] M. Herrmann, J. Müller, J. Strohbeck, and M. Buchholz, "Environment Modeling Based on Generic Infrastructure Sensor Interfaces Using a Centralized Labeled-Multi-Bernoulli Filter," *submitted to 2020 IEEE International Conference on Multisensor Fusion and Integration (MFI 2020)*, 2020.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 4510–4520.

[24] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," *arXiv preprint arXiv:1608.08242*, 2016.

[25] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[26] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 7482–7491.

[27] S. Chowdhuri, T. Pankaj, and K. Zipser, "Multinet: Multi-modal multi-task learning for autonomous driving," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Jan 2019, pp. 1496–1504.

[28] Argo AI, "Argoverse motion forecasting competition," 2019. [Online]. Available: https://evalai.cloudcv.org/web/challenges/challenge-page/454/overview

[29] A. Paszke, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[30] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[31] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, "Lookahead optimizer: k steps forward, 1 step back," *arXiv preprint arXiv:1907.08610*, 2019.

[32] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[33] D. Feng, L. Rosenbaum, C. Glaeser, F. Timm, and K. Dietmayer, "Can we trust you? on calibration of a probabilistic object detector for autonomous driving," *arXiv preprint arXiv:1909.12358*, 2019.

[34] I. Hasan, F. Setti, T. Tsesmelis, V. Belagiannis, S. Amin, A. Del Bue, M. Cristani, and F. Galasso, "Forecasting people trajectories and head poses by jointly reasoning on tracklets and vislets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[35] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab, "Robust optimization for deep regression," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2830–2838.