Conditional Vehicle Trajectories Prediction in CARLA Urban Environment

Thibault Buhet, Emilie Wirbel, Xavier Perrotton
Valeo Driving Assistance Research
34 rue Saint André, 93000 Bobigny
name.surname@valeo.com

Abstract

Imitation learning is becoming more and more successful for autonomous driving. End-to-end (raw signal to command) performs well on relatively simple tasks (lane keeping and navigation). Mid-to-mid (environment abstraction to mid-level trajectory representation) or direct perception (raw signal to performance) approaches strive to handle more complex, real life environment and tasks (e.g. complex intersection). In this work, we show that complex urban situations can be handled with raw signal input and mid-level representation. We build a hybrid end-to-mid approach predicting trajectories for neighbor vehicles and for the ego vehicle with a conditional navigation goal. We propose an original architecture inspired from social pooling LSTM taking low and mid level data as input and producing trajectories as polynomials of time. We introduce a label augmentation mechanism to get the level of generalization that is required to control a vehicle. The performance is evaluated on CARLA 0.8 benchmark, showing significant improvements over previously published state of the art.

1. Introduction

Modular pipelines [32] are the most used approach to autonomous driving. The advantage is that the modules are interpretable and relatively mature, in particular on the perception side with the success of deep learning for object detection ([13, 20] among many others). However, the complexity of the interactions in the real world causes the pipeline to be also complex, especially in the planning and decision modules. The annotations for the modules are also costly and difficult to obtain.

End-to-end imitation learning (IL) is a possible answer to these issues: one single neural network is used from the raw data input to the command output, and the ground truth command is obtained by recording an expert without additional annotation effort. Note that the expert does not necessarily have to be a human: Pan *et al*. [23] use a Model Predictive Control agent with high end sensors to train for aggressive offroad driving. One of the main difficulties of IL is that the online test distribution is not the same as the recorded ground truth. This is due in particular to error accumulation from the network, which will lead the vehicle away from an ideal trajectory, where there is almost no data recorded because an expert driver avoids these situations. Without data augmentation, the common offline evaluation metrics are not correlated to the online performance, as shown for example by Codevilla *et al*. [7] who stress the importance of data augmentation and study offline metrics. The two seminal articles for IL in autonomous driving [25, 5] propose a solution: combine side cameras with the main one to emulate lateral deviations from the road. In [33, 11] the recorded data is modified to perform *label augmentation*: data similar to failure cases is generated a posteriori, but without the need of additional sensors. This augmentation is usually necessary to use IL directly, unless some additional data is collected online iteratively to correct the failure cases, using DAgger [29] for example.

Lack of interpretability is another challenge of IL, or more generally making sure the network uses the correct information. *Direct perception* is a possible solution: instead of commands, the network predicts hand-picked parameters relevant to the driving (distance to the lines, to other vehicles), which are then fed to an independent controller [6, 1, 30]. The limitation of this approach is the necessity to choose the relevant parameters.

Another option is to use mid level interpretable data as input and output of the network: the input is coming from perception modules, the output is a mid level representation like a trajectory. ChauffeurNet [3] is an example of this *mid-to-mid* approach: road users positions, road geometry and traffic light states are used to produce vehicle trajectories and control a real vehicle. This work also demonstrates that adding high level information makes it possible to scale up in terms of complexity of the situations.

Priviledged learning is another possible way to improve the network performance by providing additional information. In that approach, the network is partly trained with an auxiliary task on a ground truth which is useful to driving,

and on the rest is only trained for IL. The goal is to leverage the fact that part of the network layers were trained to produce features which are related to helpful information. Xu *et al*. [36] use semantic segmentation on the Berkeley Deep Drive dataset as an auxiliary task, and show that it increases performance on the future ego-motion prediction tasks.

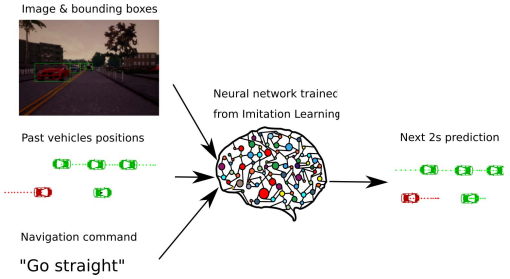

Figure 1: Inputs and outputs of the network: image, object tracks and navigation command, to predict ego and neighbors trajectories

In this article, our goal is to perform navigation tasks in CARLA [10] of an autonomous vehicle (*ego vehicle*), while also predicting the trajectories of external agents in the surroundings (*neighbors*). To do so, we propose a network taking as input: images, object detections as 3D and 2D bounding boxes and navigation commands to describe the desired behavior of the ego vehicle at intersections. It is trained to predict ego vehicle and neighbors trajectories through imitation learning, expressed as polynomials over time. This is a hybrid *end-to-mid* approach, because both raw signal and partial environment abstraction are used, and trajectories are produced. It combines the approaches of end-to-end, mid-to-mid and priviledged learning Our main contributions are:

- to propose an architecture combining raw image data and mid-level information. Our network uses contributions from pure trajectory prediction (see section 2 for more details), but also raw camera image to predict the desired output, typically for traffic lights.
- to introduce an augmentation method for these vehicle trajectories
- to use neighboring vehicles trajectory prediction as a simultaneous task.
- to perform an extensive evaluation on the CARLA 0.8 benchmark for the control of a vehicle using our trajectory prediction, with a comparison to state of the art baseline and ablation study of the augmentation and neighbors prediction

2. Related work

Conditional Imitation learning for autonomous driving Since our goal is to perform navigation but not path planning, we use navigation commands as an input. To take them into account, they can be simply concatenated in the last layers of the network [16], but this requires parameterization to give them an appropriate weight. In Conditional Imitation Learning (CIL) [8], the last layers of the network are split into branches which are masked with the current navigation command, thus allowing the network to learn specific behaviors for each goal. This method is used as is in this work because it is more scalable and interpretable. Conditional Affordance Learning (CAL) [30] adapts this to direct perception, and improves over CIL. CIL and CAL will be used as baselines for evaluation.

Simulator and benchmark A simulator is an ideal environment for IL: it is easy to generate training data, to add additional sensors for data augmentation, and to get quantitative evaluation on the online test phase. CARLA [10] is used for this article: it is dedicated to autonomous driving, with a benchmark which is used to compare to the state of the art (see section 5). Other simulators are available for autonomous driving research applications. DeepDrive [26] is a simulator originally based on the game Grand Theft Auto V (GTA V). GTA V itself has been used as a realistic simulator, but lacks flexibility for imitation learning. At the time this article is completed, a new simulator, LGSVL Simulator [18], has just been released with promising features (large environment and sensor set) but still lacks some features like autopilot for imitation learning.

Predicting trajectories and interactions from object positions Many existing contributions cover future trajectory prediction from past positions, both model based trajectory prediction [22] and with Recurrent Neural Networks (RNN) [24]. SocialLSTM [2] leverages RNNs to model interactions, here between pedestrians. SocialGAN [12] uses GAN to generate pedestrians trajectories with plausible interactions. The common idea is that interactions between agents are learnt by sharing encodings of the individual trajectories which are near to each other. The individual encodings of the trajectories are generally produced by shared layers: all agents are interchangeable. [9] shows an extension of the concept to autonomous vehicles, using a proximity map (similar to an occupancy grid) and separate trajectory encoding. Our article extends the concept of trajectory prediction with polynomials, and proves that it can be combined with image level encoding and navigation commands.

Predicting trajectories from low level data Predicting object trajectories from raw data has already received some attention. Huang *et al*. [15] perform visual path prediction from a fixed surveillance camera, to predict potential trajectories from pedestrians and cars, building a reward map of reachability. In contrast, our work relies on a moving

camera depending on the ego vehicle. [17, 28] use Lidar information in combination with image, either with imitation or model based reinforcement learning. Luo *et al*. [21] use an end-to-end network based on 3D Lidar data to track and predict future motion. We show in our work that motion forecasting can be performed with only 3D detections and 2D image information. Intention-Net [34] combines raw image with a high level plan where the desired path is drawn for indoor navigation, which shows that it is possible to take advantage of metric structure information and low level image data. This is similar to what is done in this article, but here the network itself predicts the future path.

3. Trajectory prediction from image and object detections

3.1. System description

The goal of the system is to predict the future positions on a fixed 2s horizon of the ego vehicle and the neighbors. The maximum number of predicted neighbors is fixed at $N = 5$ for simplicity. It is assumed that the past positions of the neighbors are known relatively to the current ego position (here obtained from the simulator). For the ego vehicle, the future trajectory is conditionned by a navigation command. This future trajectory can be fed to a controller, here a Proportional Integral Derivative (PID). All experiments described in this paper are conducted in CARLA, but could be extended to any system providing object tracks in 3D with a 2D reprojection, raw image input and ego vehicle odometry. CARLA autopilot is used as the expert.

To describe the trajectory, we represent positions of the object as tuples $(x(t), y(t))$ for time t, corresponding to the center of the 3D bounding boxes for the neighbors. To describe these efficiently, we chose to use a polynomial representation. For each predicted trajectory, we predict the vector $(x_0, ..., x_4, y_0, ..., y_4)$ where

$$x(t) = x_0t^4 + x_1t^3 + x_2t^2 + x_3t + x_4$$
$$y(t) = y_0t^4 + y_1t^3 + y_2t^2 + y_3t + y_4$$

Note that the coefficients are proportional to speed, acceleration, jerk and derivative of jerk along the x and y axis. For the other vehicles, the trajectory is given relatively to their current position. This makes it possible to predict vehicle trajectories independently of their distance to the ego vehicle. This is a valid assumption because these shifted trajectories have the same shape as the original ones, and the influence of the relative vehicle positions is still encoded in the proximity map (see Section 3.2 for details). This is also a way to get more relevant training data and generalization, because all inputs are in a similar domain.

3.2. Network structure

The following data is fed as input to the network:

- a front facing camera RGB image I ($320 \times 240 \times 3$)
- the past positions of the ego vehicle and neighbors, sampled every $\delta t = 0.1s$ for the last 2s, which corresponds to $T = 20$ values. They are given in the reference of the current ego vehicle position. They are grouped using a sliding window of size $K = 3$. These vectors are noted V_i for neighbors (with $i \in [0, N[$) and E for the ego vehicle.
- a navigation command nc. If we are approaching an intersection, the goal can be *left*, *right* or *cross*, else the goal is *keep lane*.

V_i and E are projected into a proximity map M. This map represents a local neighborhood of the current ego vehicle position, of size 65×10.5m, with the current ego at the center. The dimensions of the area around the ego, used in [9], corresponds to 3 standard lanes side by side for around 2s front and rear at 50kph. We use a coarse resolution (13×3) to represent this neighborhood as an array where each cell is roughly the size of a vehicle. Each cell contains K=3 consecutive positions ending at the corresponding instant, if there is a vehicle at that position and at that time, and nothing elsewise. We stack this representation for the past T frames, resulting in a $13 \times 3 \times T \times 2 \times K$ array. Note that the information of V_i is contained in M: we distinguish the two to be able to take an arbitrary neighbors into account for the context, but still be able to predict the future of K chosen tracks. The past 3D bounding boxes of the road users in the current reference are projected back in the current camera space. This gives a $320 \times 240 \times K$ array B, where the last dimensions contains the bounding box at t=0 plus the 2 previous ones.

The full structure of the network is described in Figure 2. The global context from both image and proximity map is encoded into two feature vectors, concatenated to form a context feature vector C. The image concactenated with the neighbors bounding boxes is encoded using a VGG16 [31] network structure in which we replaced the first convolutional layer by a CoordConv [19] layer with the same parameters, followed by a global max pooling layer to make the feature vector dimension independent from the input dimension. The VGG16 could be replaced by any other image encoder, but makes visual backpropagation [4] easier. The proximity map is encoded using a Convolutional Long Short Term Memory layer (ConvLSTM) [35], followed by a convolution and a max pool. Note that here the proximity maps are treated as images and not as a time serie.

In parallel, the ego vehicle trajectory and the other vehicles trajectories are encoded, with a fully connected layer

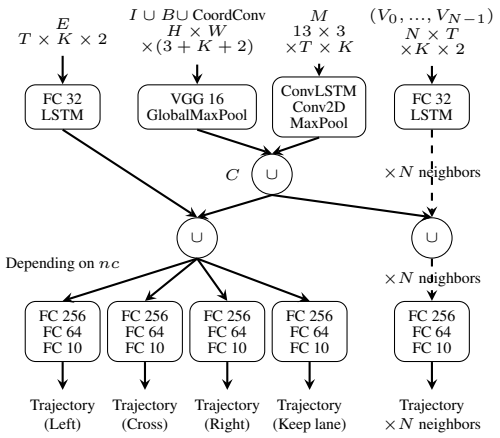

Figure 2: Trajectory prediction network (FC X= fully connected with X outputs, Conv2D=regular convolution, $\cup$=Concatenation, see section 3.2 for other notations)

followed by an LSTM [14]. The neighbors encoding are N twin layers with shared weights. The trajectory encodings are concatenated with the context encoding, then fed into fully connected layers to finally produce the polynom coefficients. The neighbor weights are shared between neighbors and separate from the ego weights.

Finally, the context encoding is concatenated with the ego encoding, then fed into branches corresponding to the different high level goals. A mask is used to choose the branch corresponding to the current high level goal. Neighbors encodings are concatenated with the context encodings then decoded in parallel, with shared weights.

As stated in the introduction, this network structure is a significant contribution on pre-existing work [8, 9]. Part of it relies on the principles of Social LSTM and the proximity map of [9], but with an original contribution on the trajectories decoding, the final output format and the distinction between the controlled ego vehicle and the neighbors. It also introduces the concept of global image context encoding concatenation, which is then decoded individually.

4. Training the network

4.1. Ground truth and loss

To generate the ground truth, scenarios are recorded with different vehicles driving around the CARLA Town 01, one of them being the ego vehicle. Once the data is recorded, training data is extracted by using a sliding window of size 4s: 2s for the past, and 2s for the anticipation. On each of these windows, the positions of the ego vehicle and its neighbors are computed relatively to the middle of the window. This makes it possible to generate E, M and $(V_0, ..., V_{N-1})$. The input image I is the camera frame at the middle of the time window. All positions of the neighbors vehicles are projected back into I, using the camera calibration, so that B can be generated. Least square polynomial fit is used to get the coefficients for the future trajectory, based on the points from the second half of the window. The navigation command is computed offline using the recorded positions and orientation of the ego vehicle and a list of positions of all the intersections in the map.

The loss is not done directly on the estimated polynomials coefficients, to be independent from the polynom degree and the degree of the coefficient. For example, an error on coefficient x_0 has much more impact than on x_4. Instead, 2D points of the fixed anticipation are sampled with the time step δt, which are noted e_i and $\hat{e}_i$ with $i \in [1, T]$ for the ego vehicle ground truth and prediction respectively, and $v_{k,i}$ and $\hat{v}_{k,i}$ for the k-th neighbor ground truth and prediction respectively. We then use a L2 loss on these points:

$$L(\hat{e}_i, e_i, \hat{v_{k,i}}, v_{k,i}) = \sum_{i=1}^{T} ||\hat{e}_i - e_i||^2 + \sum_{k=0}^{N} \sum_{i=1}^{T} ||\hat{v}_{k,i} - v_{k,i}||^2$$

For training, we used Adam optimizer with a starting learning rate of 10^{-5} and a batch size of 8.

4.2. Label augmentation

The main challenge of IL is the difference between train and online test distribution, in particular since the ego vehicle is controlled with the trajectory prediction. To reach the acceptable level of performance, we combine classical randomization with label augmentation, which means generating new inputs and labels from existing data.

Randomization is added first to ensure a better generalization. To increase the variability in the training data, we successively recorded 3 minutes episodes with randomized parameters (40-80 vehicles, 10-30 pedestrians, starting positions). In addition, all vehicles models and pedestrians models are chosen randomly, and the paths of the vehicles and the pedestrians are also random. The map used, Town 01, is fixed by the benchmark training conditions and the weather is random between the 4 weathers allowed for training: clear noon, wet noon, hard rain noon, clear sunset.

Noise is added to the input observations, to avoid overfitting, ensure better generalization but also emulate noisy data that could be obtained from a real environment. It also prevents the network from learning to simply extrapolate from the past trajectory, as noted for example in [3]. A Gaussian noise is added to the past positions and image bounding boxes of the vehicles. In parallel, vehicles are randomly removed from or added to the proximity map.

As expected, a training without label augmentation to prepare online behavior yields poor results because of error accumulation. After training and testing online without label augmentation three main problematic behaviors were

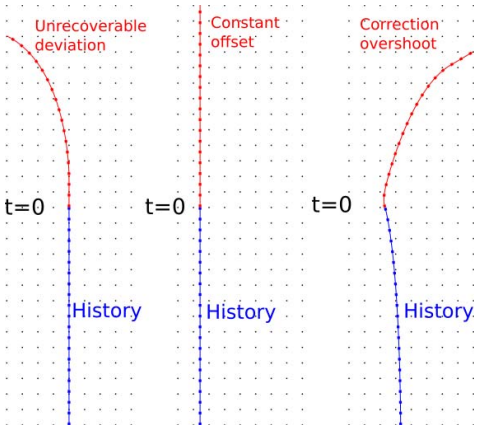

Figure 3: Example of problematic behavior produced without label augmentation: unrecoverable deviation (left), lateral offset (center) and recovery overshoot (right)

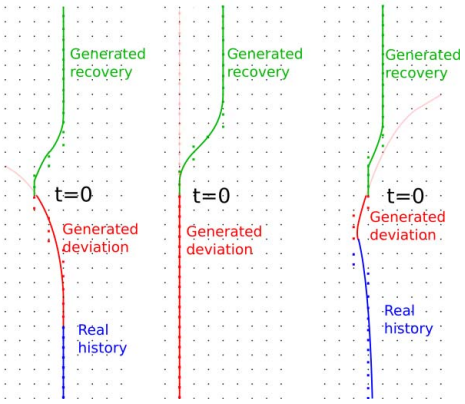

Figure 4: Example of label augmentation for incorrect behavior correction: large error recovery (left), lane recentering (center) and overshoot compensation (right)

observed (represented in Figure 3): *unrecoverable deviation*, *lateral offset* (the prediction has a constant lateral offset relatively to the ground truth), and *recovery overshoot* (the prediction over-compensates after a deviation).

To correct these behaviors, we create a randomized artificial history for each and generate an artificial correct future. To do so, a lateral deviation and/or an orientation deviation are added to the trajectory at some time in the past history, with a varying amplitude, deviation and recovery time. The deviation is always introduced in the past, since the network should not be trained to induce deviations but to recover from them. To get the corresponding input image, cameras are added and recorded to emulate positions of the ego vehicle with lateral and angular offset. We choose to add 4 lateral positions every 0.2 meters on each side of the car with 3 possible orientations for the cameras: facing front, facing a point 10 meters in front of the car and the symmetric. Figure 4 illustrates the simulated deviations and the corresponding recoveries, from an existing trajectory. Results from Section 5 show that even if the model is simple, this label augmentation brings considerable robustness to the system at online test time.

In the end, all recordings joined together sums up to 15 hours of ego vehicle driving, strictly respecting the benchmark training conditions (predetermined weathers and map). About 20% of the episodes were recorded with the ego car equipped with the 25 cameras, and all the tracks of these episodes are artificially deviated in the final dataset. The dataset on Town 01 is split into two parts only: training (90%) and validation (10%), testing is done online in the simulator on both maps of CARLA. We chose not to record any data on Town 02 map, even for validation of the models so it remains a pure test for the trajectory prediction.

5. Results

5.1. Metrics

The CARLA benchmark (first introduced in [8]) is used to quantify the network performance in terms of ego vehicle trajectory prediction, and compare it to the state of the art. The benchmark regroups navigation tasks of increasing difficulty: going in a straight line, taking one turn, and navigation task with several turns without or with other road users. It can be performed on the training town (Town 01) or on the test town (Town 02). The benchmarks metrics record the percentage of successfully performed tasks, which means that the vehicle reached its destination, and the distance between two infractions (driving on the wrong side of the lane, hitting something etc). In accordance with the state of the art reported results, we compare the results for Town 01 and train weathers (*training*), Town 01 and test weathers (*new weather*), Town 02 and train weathers (*new town*) and Town 02 and test weathers (*new town + weather*). Even if the benchmark takes place in the town where the training data is recorded, this is not the training data: not only this is an online test, but also the input data is not the same.

Note that the default timeout before considering an experiment failed is multiplied by 3. This is necessary because the default is calculated based on the total path distance and since our network must stop at red lights, this drastically reduces the average mean speed of the controlled car, especially in Town 02 where traffic lights can be very close to each other. This adaptation should not impact the benchmark results, the average speed while not stopped is close to the urban driving speed of Carla's autopilot (20kph).

To provide quantitative results on the neighbors trajectories prediction, we provide metrics on the trajectories from our test set. Mean Average Error (MAE) between predicted and ground truth trajectory points is used to quantify the

Task	Training			New weather			New town			New town + weather		
	CIL	CAL	Ours	CIL	CAL	Ours	CIL	CAL	Ours	CIL	CAL	Ours
Straight	95	**100**	**100**	98	**100**	**100**	**97**	93	**97**	80	94	**98**
One turn	89	97	**100**	90	96	**100**	59	82	**87**	48	72	**90**
Navigation	86	92	**99**	84	90	**100**	40	**70**	58	44	68	**74**
Nav. dynamic	83	83	**98**	82	82	**100**	38	64	**68**	42	64	**82**

Table 1: Success rate comparison (in % for each task and scenario, more is better) with baselines [8, 30]

Infraction type	Training			New weather			New town			New town + weather		
	CIL	CAL	Ours	CIL	CAL	Ours	CIL	CAL	Ours	CIL	CAL	Ours
Opposite lane	33.4	6.7	**75.45**	57.3	$>$ **60**	$>$**35.7**	1.12	**2.21**	1.45	0.78	**2.34**	1.24
Sidewalk	12.9	6.1	**37.73**	$>$**57**	6.0	$>$**35.7**	0.76	0.88	**1.53**	0.81	1.34	**2.01**
Collision: static	**5.38**	2.5	3.97	4.05	6.0	**11.9**	0.40	0.36	**0.46**	0.28	0.31	**0.44**
Collision: car	3.26	12.1	$>$**75**	1.86	$>$**60**	$>$**35.7**	0.59	2.04	**5.12**	0.44	1.38	**16.15**
Collision: ped.	6.35	**30.3**	15.09	11.2	$>$**60**	8.93	1.88	**26.49**	3.25	1.41	**6.72**	2.31

Table 2: Infraction distance comparison (in km between infractions, more is better), with baselines [8, 30]. Distances with $>$ indicate that no infraction was encountered during the whole benchmark.

prediction error. The MAE is computed on the whole future trajectory, but also measured for the specific points at t=2s. The goal is to differentiate between global error, short and long term prediction. The same metrics are also provided offline for the ego vehicle prediction.

5.2. Benchmark quantitative results

We compare our results to two different baselines: Conditional Imitation Learning (CIL) [8] and Conditional Affordance Learning (CAL) [30]. Both have a structure similar to our implementation, except that CIL produces instantaneous commands and CAL produces affordances which are then given to a controller. To control the car in the simulation, a simple PID controller was used to transform trajectory predicted by the network into a car command. The speed is capped to 20kph maximum, and the control could probably be improved (filtering or smoothing), which could lead to a slight improvement on the results.

Table 1 summarizes the comparison for success rates on the benchmark tasks. Our method outperforms CIL on practically all tasks and all conditions, and the difference is more significant for tasks harder than the straight line. We are competitive with CAL: we equal or outperform on training conditions, and outperform CAL on the most general test conditions. We note that for the hardest tasks, dynamic navigation in new town, our method performs better when other road users are present in the simulation. In fact this situation is closer to the training data and the network must have learned to partially rely on the other vehicles position to predict trajectories. Our method also outperforms CIL, except for the static collision infraction. The gap is especially visible in tasks related to steering (opposite lane and sidewalk). This is probably due to the label augmentation which is designed to keep the vehicle in the nominal trajectory. For test cases on car collision, our algorithm also performs very well, which is certainly due to the neighbors positions input. Note that our approach is significantly weaker for pedestrian infractions, even if it still is comparable to CIL. This is probably linked to the fact that there were few pedestrians in our training base.

	Training	New weather	New town	New town + weather
Ratio (%)	3.0	1.8	21.9	25.3
Total	436	436	283	170

Table 3: Evaluation of the number of red light runs of the network: percentage of red lights run over all encountered traffic lights count (dynamic navigation task of the benchmark, lower is better)

Table 3 shows a quantitative evaluation of the traffic lights infractions. We report the ratio of red lights run over all encountered traffic lights for the dynamic navigation task. Note that this number is affected by the traffic light states, because the network cannot run a light that is green, and the states encountered vary with the vehicle speed, starting positions, traffic state etc.

5.3. Ablation studies

The impact of our data augmentation is measured in Table 4. Three scenarios are compared, in Town 01 : no la-

bel augmentation (only randomization is introduced), partial label augmentation (performed on only one of the 4 train weathers), and full (performed on all train weathers). Table 5 compares the MAE obtained on Town 01 train and validation data weathers.

	Train			Validation		
Augmentation	None	Part	Full	None	Part	Full
Straight	70	100	100	54	100	100
One turn	21	100	100	14	100	100
Navigation	16	93	99	12	96	100
Nav. dynamic	14	97	98	4	100	100

Table 4: Comparative success rates (in %) without, with partial and full data augmentation in Town 01 (None, Partial, Full) for train and validation weathers

The comparison in Table 4 confirms that label augmentation is critical to the performance of the vehicle control: there is a significant gap when introducing the augmentation, then a slight improvement when we deploy it on all conditions. It is interesting to note that the effect is much more noticeable on complex navigation tasks, where the maneuvers are more complex so that errors accumulate quicker. However, the offline quantitative evaluations of the MAE from Table 5 are almost identical. This proves that online test is the real significant indicator for IL when it is used for active control. Note that this is in line with the findings of [7], which highlights that the correlation between offline metrics and online performance is weak.

Weathers	Train			Validation		
Augmentation	None	Part	Full	None	Part	Full
Ego	**0.05**	0.06	0.06	**0.09**	**0.09**	0.10
Ego 2s	**0.11**	0.12	0.13	0.23	0.23	0.23
Neighbors	0.10	0.10	0.10	0.23	0.22	**0.21**
Neighbors 2s	0.22	0.22	**0.21**	0.54	0.54	**0.51**

Table 5: Comparison of MAE on train and validation data (in m), with none, partial and full data augmentation (None, Partial, Full), less is better

Table 5 also shows that the error is greater for the neighbors than for the ego. This is not surprising, because the ego trajectory has the additional information of the navigation command, whereas neighbor trajectories can be ambiguous at intersections for a limited time, in which case there is an error on the future trajectory. In the case of neighbours prediction, the measure is more relevant because there is no online control loop for the neighbors.

Table 6 illustrates the influence of the neighbors trajectory prediction on the global performance. In this ablation, the loss on neighbors is removed, but the proximity map is kept, so there is no backpropagation on the neighbors encoding and decodings. Very interestingly, predicting the neighbors actually reduces the percentage of tasks completed, but performs significantly worse for all infractions except static collisions. It appears that adding the neighbors prediction makes the ego prediction more compliant to traffic rules. This opens new research directions to investigate how mixing neighbors and ego trajectory prediction impacts performances.

	Train		Validation	
With neighbors	Yes	No	Yes	No
Navigation (%)	58	**84**	74	**92**
Nav. dynamic	68	**81**	82	**92**
Opposite lane (km)	**1.45**	0.62	**1.24**	0.19
Sidewalk	**1.53**	0.32	**2.01**	0.16
Static	0.46	**1.26**	0.44	**0.60**
Car	**5.12**	0.25	**16.15**	0.19
Pedestrian	**3.25**	0.25	**2.31**	0.25

Table 6: Ablation study of the neighbor prediction for Town 02 train and validation weathers (for navigation tasks)

None		Medium		High		Neighbors		YOLO	
train	test	train	test	train	test	train	test	train	test
68	82	69	80	63	76	74	74	67	78
3D bounding box noise standard deviation (m)									
x	y	x	y	x	y	x	y	(Axis)	
0.0	0.0	0.1	0.05	0.3	0.15	0.1	0.05	(Ego)	
0.0	0.0	0.1	0.05	0.3	0.15	1.0	0.5	(Neighb.)	

Table 7: % success rate for navigation with varying noise (train & test weathers) and YOLO images bounding boxes

A final ablation study has been done on the influence of noise on vehicle positions. In this all other evaluations, the ground truth positions are used, but this is not available in the real world and could constitute a bias in the comparison with state of the art. To prove the robustness, a Gaussian white noise of increasing deviation is added to the positions of the objects, both at training and testing time. In parallel, the image bounding boxes, which originally are a projection of the objects positions in the image, are replaced by the output of a YOLO [27] detector trained on CARLA images. For time constraints, the study is conducted only for the success rate of the dynamic navigation task. Table 7 illustrates this: even for high levels of noise and using an image detector, the performance drop is limited. This proves that any reasonably robust object detector could be used instead of the ground truth position.

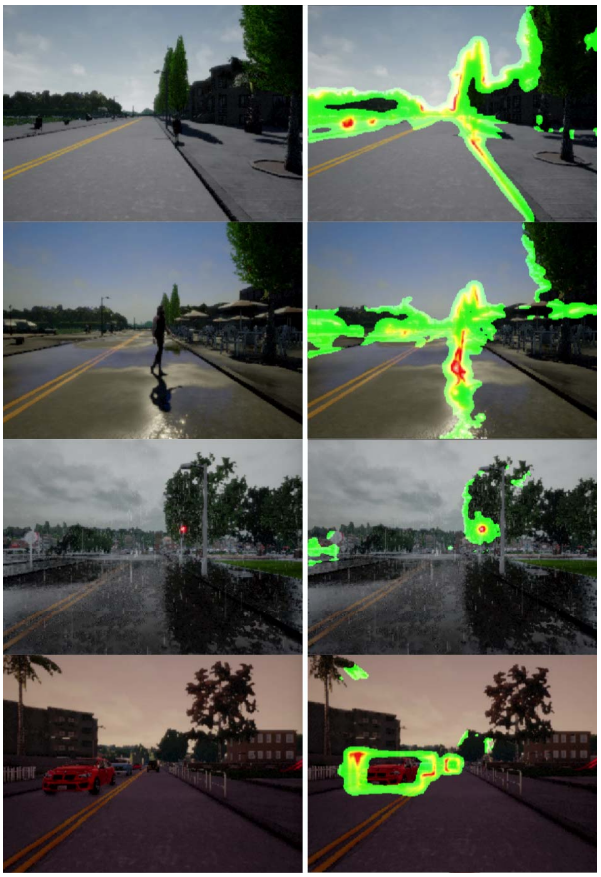

Figure 5: VisualBackprop (right) for ego trajectory prediction and corresponding image (left), best viewed in color. The heatmap overlay represents area of activation(green is low and red is high). From top to bottom: ego car too close to the sidewalk, pedestrian crossing, before stopping at red light, dense traffic

5.4. Qualitative evaluation

To qualitatively explain the outputs of the network, we rely on VisualBackProp [4], which highlights the image pixels which contributed the most to the final results. Figure 5 shows typical examples of the backpropagation. Vehicles are predominant, which is not suprising because their boxes are provided as input. More interestingly, the VisualBackprop also highlights lane markings and curbs which are relevant for lateral positioning, but mostly when there is a significant lateral deviation. Traffic lights with the current color light and pedestrians are also highlighted when relevant. This is consistent with the fact that this information can only be found in the image, and not in the position history, and shows that it is possible to learn a link between the image and the metric space.

The autopilot in CARLA respects traffic rules, in particular traffic lights. They are taken into account by the trained network, not only for the ego vehicle, but also for the neighbors. A typical situation is a queue at a red light, with at least one vehicle standing before the ego vehicle. While the light is red, all vehicles are predicted stopped. When the light turns green, the trajectory of the first vehicle in the line predicts a restart while the others are still stopped. Then the vehicles are predicted to restart progressively, one after the other. Figure 6 is a typical example of how position and image information are correlated, and how the networks takes vehicle interactions into account.

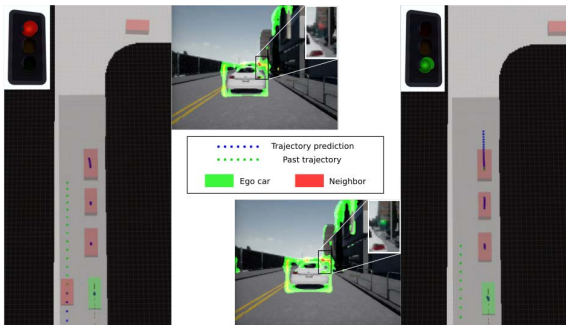

Figure 6: Image and predicted trajectories at red light (top), right after light turns green (bottom). The vehicle at the front of the queue has a restart trajectory predicted, but the ego vehicle is still predicted to be stopped for the next 2s.

The supplementary material contains a video of the network driving around CARLA Town 01, with the VisualBackprop and the vehicles trajectory predictions (on top view and projected on the ego vehicle frontal camera).

6. Conclusion

In this work, we have proposed a hybrid end-to-mid neural network to predict vehicle trajectories in CARLA urban simulated environment. The network integrates positional and image information, learns vehicle interactions and extracts relevant data from image. It is applied to a navigation scenario, and produces significant improvement over previous state of the art on the CARLA benchmark. This shows that a mix of high and low level data, together with auxiliary tasks, bring performance to imitation learning. This work also highlights the impact of label augmentation: adding artificial data helps reduce the gap between train and test distribution and increases the performance drastically.

In future work, this framework could be applied on real data. To make the network truly end-to-mid, the object detection input could be integrated inside the network, with an auxiliary goal to be trained on object detection or a similar task. Investigations on the influence of additional predictions would bring insights for the structuring of end-to-end networks.

References

[1] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha. Deep learning algorithm for autonomous driving using googlenet. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 89–96. IEEE, 2017. 1

[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016. 2

[3] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 1, 4

[4] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. J. Ackel, U. Muller, P. Yeres, and K. Zieba. Visualbackprop: Efficient visualization of cnns for autonomous driving. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 3, 8

[5] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 1

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015. 1

[7] F. Codevilla, A. M. López, V. Koltun, and A. Dosovitskiy. On offline evaluation of vision-based driving models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 236–251, 2018. 1, 7

[8] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. 2, 4, 5, 6

[9] N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. *CVPR TrajNet Workshop*, 2018. 2, 3, 4

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 2

[11] L. George, T. Buhet, E. Wirbel, G. Le-Gall, and X. Perrotton. Imitation learning for end to end vehicle longitudinal control with forward camera. *arXiv preprint arXiv:1812.05841*, 2018. 1

[12] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 2

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 1

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4

[15] S. Huang, X. Li, Z. Zhang, Z. He, F. Wu, W. Liu, J. Tang, and Y. Zhuang. Deep learning driven visual path prediction from a single image. *IEEE Transactions on Image Processing*, 25(12):5892–5904, 2016. 2

[16] C. Hubschneider, A. Bauer, M. Weber, and J. M. Zöllner. Adding navigation to the equation: Turning decisions for end-to-end vehicle control. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 1–8. IEEE, 2017. 2

[17] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 3

[18] LGSVL. Lgsvl simulator. https://www.lgsvlsimulator.com/. 2

[19] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9628–9639, 2018. 3

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[21] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 3

[22] H. Mouhagir, V. Cherfaoui, R. Talj, F. Aioun, and F. Guillemard. Using evidential occupancy grid for vehicle trajectory planning under uncertainty with tentacles. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pages 1–7. IEEE, 2017. 2

[23] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots. Agile autonomous driving via end-to-end deep imitation learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018. 1

[24] S. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. *arXiv preprint arXiv:1802.06338*, 2018. 2

[25] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 1

[26] C. Quiter and M. Ernst. Deepdrive 2.0. https://deepdrive.io/, Mar. 2018. 2

[27] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 7

[28] N. Rhinehart, R. McAllister, and S. Levine. Deep imitative models for flexible inference, planning, and control. *arXiv preprint arXiv:1810.06544*, 2018. 3

[29] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011. 1

[30] A. Sauer, N. Savinov, and A. Geiger. Conditional affordance learning for driving in urban environments. *CoRR*, abs/1806.06498, 2018. 1, 2, 6

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[32] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006. 1

[33] M. Toromanoff, E. Wirbel, F. Wilhelm, C. Vejarano, X. Perrotton, and F. Moutarde. End to end vehicle lateral control using a single fisheye camera. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3613–3619. IEEE, 2018. 1

[34] G. Wei, D. Hus, W. S. Lee, S. Shen, and K. Subramanian. Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation. *arXiv preprint arXiv:1710.05627*, 2017. 3

[35] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015. 3

[36] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3530–3538. IEEE, 2017. 2