Learning to Predict Motion from Raw 3D Object Detections

Neumeyer, C.; Bijelic, Mario ; Gavrila, D.

DOI
10.1109/IV51971.2022.9827071

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IEEE Intelligent Vehicles Symposium (IV)

2022 IEEE Intelligent Vehicles Symposium (IV)
June 5-9, 2022. Aachen, Germany

Learning to Predict Motion from Raw 3D Object Detections

Christian Neumeyer[1,2], Mario Bijelic[3], Dariu M. Gavrila[1]

***Abstract*— We show how to design a motion prediction algorithm that works with 3D object detections and map locations. In particular, we obtain object id's – even though the training data does not contain any object id's – across multiple time-steps into the future by propagating a Gaussian Mixture of likely object (e.g., vehicle) locations through time.**

We validate our approach on the nuScenes dataset. First, we find that a motion prediction algorithm without tracking id's performs as well as motion prediction algorithm with tracking id's in the training data. Second, the 3D labels of an on-board perception system are inferior (e.g., loss of detections, positional uncertainty) to those generated by offline labelling (automatic labelling pipeline, manual labelling). Even so, we find that a moderate increase in the size of the training data offsets the deterioration in prediction performance (with no additional offline labelling).

2022 IEEE Intelligent Vehicles Symposium (IV) | 978-1-6654-8821-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IV51971.2022.9827071

I. INTRODUCTION

Autonomous vehicles need to anticipate what other traffic participants will do in order to safely and efficiently navigate in complex urban environments. Challenges include other vehicles taking turns and pedestrians stepping on the street. Companies and research institutions published multiple data sets to address this challenge. nuPlan [1] is the most extensive publicly available data set containing 0.2 years of driving data. This is still a fraction of the driving time of the cars produced by any major OEM on any given day.

Deep learning algorithms are the state-of-the-art in human motion prediction and benefit from ever more data. It is likely that privately owned vehicles (not research vehicles) will provide the bulk of the training data, with Tesla being an early example. However, raw sensor data (e.g., camera) is large compared to, e.g., raw 3D object detections (3d-position and orientation). One would also need to label the raw sensor data, the costs of which easily exceed those of training a neural network. Collecting raw sensor data might also violate the privacy of the vehicle owner and bystanders. Finally, a domain gap is introduced when training the prediction algorithm on perfectly labelled data that it will not get as an input in the real world.

We cannot say for sure which sensor setups will dominate commercial autonomous vehicles: Waymo currently pursues a fusion of multiple sensor types, while Tesla believes mono-camera input is all we need. Therefore, we want a representation that is agnostic to the type of sensor used, be it lidar, stereo-camera, mono-camera, radar or any combination of these. An intuitive choice is the output of the on-board perception system (e.g., 3D object detector only or 3D object detector+tracker), i.e., the three-dimensional position (and orientation) of an object (car, pedestrian). Two advantages stick out: The representation is data-efficient, and it respects the privacy of others.

Transferring and processing multiple GB of data from each privately owned car every day would be unacceptable. So, how much can we expect? We consider 2.000.000 cars – close to the annual production of Mercedes-Benz cars – where every car runs detection algorithms in the background even if no autonomous systems are engaged. A quick back-on-the-envelope calculation tells us how much data these vehicles may stream back to Mercedes-Benz every day. We assume 43 minutes of average driving [2] (Germany), around ten objects in each frame, ten frames a second, 3-d position, orientation for each object and location coordinates of the Mercedes-Benz car. We arrive at around 16.5 TB of data which amounts to 8.25 MB per vehicle each day. The total equates to 163 years of driving compared to 0.2 years of driving for the most extensive public driving data-set (nuPlan [1]) – every day. Also, the training data is identical to the data that the prediction algorithm sees when deployed in the real world. It is the same on-board perception system that we use in production cars that generates the labels of the training data-set. There is no domain gap.

One major drawback remains: The labels from an on-board perception system are noisy while existing work [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14] considers data-sets with near-perfect labels. We will address how to leverage noisy data from an on-board perception system to learn to predict the future motion of traffic participants. We show how to adapt a convolutional neural network to the task and analyse the performance gap between high-fidelity labels and those from an on-board perception system.

II. RELATED WORK

Human motion prediction received much attention in recent years. Examples include predicting a cyclist turning left [15], a pedestrian stepping on the street [16] and the motion of a vehicle [6]. [17] provides a good overview of human motion prediction.

We can distinguish between an object-centric approach to human motion prediction and object-agnostic approaches. Object-centric methods predict, e.g., the 3D position of a vehicle, while object-agnostic predict, e.g., raw sensor data with no notion of individual objects. Examples of the former include methods based on convolutional neural networks [3], [4], [5], [6], [7], [8], [9], [10], [14], graph neural networks [11], [12] and PointNet/Voxels [13]. The latter considers the

[1]Intelligent Vehicles Group, TU Delft, The Netherlands
[2]Environment Perception Group, Mercedes-Benz AG, Germany
[3]Computational Imaging Lab, Princeton University, United States of America

evolution of an occupancy grid map (with or without semantic information), e.g., [18], [19] or the evolution of sensor data, e.g., [20].

Object-agnostic methods appear promising – occupancy grid maps do not need any labelling for the training data-set. They use a high-level representation of the environment – though they lack major benchmarks. At the same time, there is a wealth of benchmarks for object-centric methods. The performance of object-agnostic methods on long-term prediction tasks (4+ seconds) compared to object-centric methods is not well established either.

Object-centric trajectory prediction usually follows the detection-tracking-prediction paradigm, where we have three separate modules that need to be optimized individually. [11], [20], [21] recently challenged that paradigm with empiric evidence that small amounts of noise in the tracking algorithm can severely impact the prediction module during deployment [21]. They propose to get rid of tracking as an intermediate step for the prediction task. They compare different convolutional neural network based architectures that either use tracking id's as an input or not. It is important to note that their tracking-free convolutional neural network still uses track ids during training to calculate the prediction loss. [11] show state-of-the-art performance on tracking and motion prediction when optimizing both on a shared neural network backbone. They execute the tracking and prediction task in parallel. Thereby, they eliminate the sequential order of tracking and then prediction. During training, tracking id's are essential to get prediction and tracking to work. Also, they use recurrent neural networks in the network architecture. It is unclear how we could employ a similar architecture where tracking ids are missing during training. [20] invert the traditional pipeline by starting with the prediction step (forecasting a lidar point cloud) and then detecting and tracking objects. While the results are not state-of-the-art, the approach appears to work well.

Recently, research on auto-labelling pipelines, e.g., [22], [23], weakly-, e.g., [24] and self-supervised algorithms, e.g., [25] for object detection intensified to suppress the need for massive human labelling efforts that cause significant delays between data gathering and training, potential quality issues with those labels, while also incurring high costs. While auto-labelling does reduce the labelling effort significantly, the initial setup and maintenance absorb engineering talent and add complexity and costs to the development of autonomous vehicle software. Also, just like manual labelling, we need to collect raw sensor data.

We can use various methods to predict the behaviour of humans in traffic situations. Not all of these apply directly to the case where the tracking-id of an object is unreliable and where we encounter false positive and false negative detections. We consider a recurrent neural network as an example. At each time step, it consumes the current position of an object and integrates that into its hidden state. The fusion of these hidden states happens at a later stage (using, e.g., social pooling [26] or graph neural networks [27]). We need to know the identity of every object through time. Otherwise, we cannot assign the correct object to its hidden state. We consider a scenario where the identities are unknown (we use pure 3D object detections) or uncertain (output of an on-board perception system).

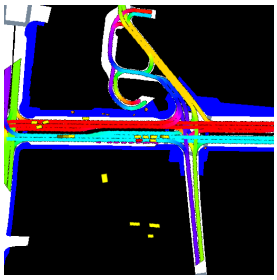

Fig. 1. We used the nuScenes dev-kit to create this image. The convolutional neural network receives a 512 x 512 pixel (240 meters x 240 meters) rasterized image as an input. The image is centred at the data-capturing vehicle (red rectangle) at $t = 0$ and includes a 120-meter radius. The vehicles are yellow rectangles with their histories (one second into the past at 2Hz, i.e., two time-steps) as yellow rectangles with a darker shade. Orange blobs correspond to pedestrians. The colours of the lanes indicate the driving direction, and walkways are coloured in blue. The only information needed to create this image is the location of the data-capturing vehicle and the 3D locations (including yaw) of other vehicles/ pedestrians and cyclists.

Convolution neural networks, on the other hand, do not need tracking id's in the input [21]. Multiple authors apply convolutional neural networks to the motion prediction task with great success [3], [4], [5], [6], [7], [8], [9], [10], [14]. While the input can be agnostic to the tracking id, the output still relies on an association between the prediction and the ground truth. We do not know the association, so it is unclear which ground-truth detections our prediction corresponds to.

Contributions

- We present a convolutional neural network motion prediction algorithm that can use noisy 3D object labels (location, orientation) during training and inference. We do not provide any tracking ids during training or inference. Nevertheless, the motion prediction algorithm can reason about specific objects (i.e., tracking id's) and their evolution through time.
- We show that the tracking id in the loss function may not be relevant to the prediction performance
- We analyse how detector noise (loss of labels, positional noise) will affect the prediction performance and give an intuition of how much data we need to compensate for that

III. PREDICTION WITHOUT TRACKING

We will now design a neural network architecture and a loss function for learning predictions without any tracking id's. First, we discuss an architecture and loss function that uses tracking id's and that is similar to, e.g., [21]. We do not claim any novelty on the setup. However, specifics like using Unet as a backbone [28] may be unique. We use the Unet [28] architecture that we truncate down to the second deconvolutional layer, and then we add a 1x1 convolution on top. In line with existing approaches such as [3], [5], [6], [7], [8] we convert the high-definition map into a colourized

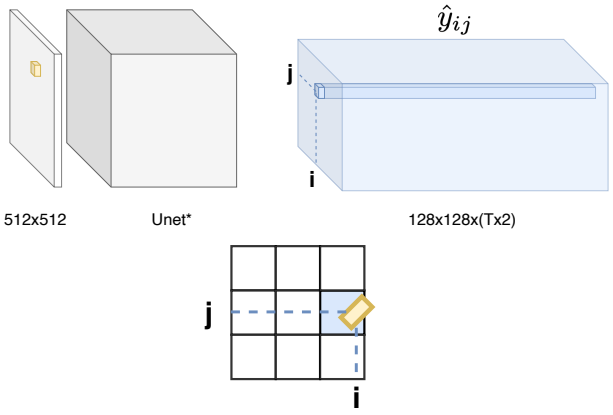

Fig. 2. Motion prediction **with** tracking id. A Unet* (truncated to 128x128 output layer) receives the 512 x 512 input (240 m x 240 m). We take a known object (yellow box) and identify which of the 128 x 128 grid cells it corresponds to. Then we obtain the prediction $\hat{y}_{ij}$ for the next T time-steps.

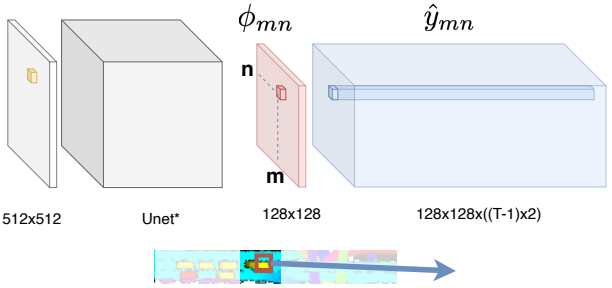

Fig. 3. Motion prediction **without** tracking id. A Unet* (truncated to 128x128 output layer) receives the 512 x 512 input (240 m x 240 m). It predicts $\hat{y}_{mn}$, the likely location of objects at time-step $t+1$ (red), in a 128 x 128 grid of Gaussian Mixture modes with weights ϕ_{mn}. Additionally, we predict the mean shift for the m, n mode in the 128 x 128 grid and propagate it through time (blue). The (cropped) image at the bottom illustrates the result. The neural network identifies objects (red rectangle) and predicts their motion (blue arrow).

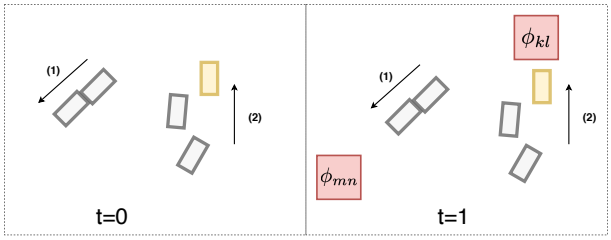

Fig. 4. $t = 0$: Two vehicles move from the top down (1) and the bottom up (2). The grey boxes indicate past detections; the yellow box indicates a current detection (at $t = 0$). Our prediction algorithm receives past detections and the detections at $t = 0$. It needs to identify the object locations during training, but we cannot use the $t = 0$ detections as ground-truth labels as they may be missing (1) or exhibit positional noise (2). Instead, we predict detections at $t = 1$ (red) that are not part of the input. This forces the neural network to reason about false negatives, object identities and positional noise, i.e., the task that a tracking module usually performs. Typical on-board perception systems run at 10 Hz so that the predictions at $t = 1$ will be close to the ground-truth position at $t = 0$.

image where colours have semantic meaning. Additionally, we visualize the detections into the image like [5], [6], [7]. Figure 1 gives an example of the input. The convolutional neural network encodes the rasterized image of size 512 x 512 and then decodes it again to a tensor of size 128 x 128 x T x 2 – the predictions of all objects in the rasterized image. Each grid cell ij contains the uni-modal prediction $\hat{y}_{ij} = [\hat{y}_{ij0}, ..., \hat{y}_{ijT}]$ of the position over a time-horizon of T time-steps for an object in that grid cell (see figure 2). Note that we know the ground-truth $y = [y_1, ..., y_T]$ of every object from one second of the past to six seconds into the future (e.g., manual labels).

We can optimize the neural network with the following l2 loss function.

$$L = \frac{1}{T} \sum_{t=1}^{T} ||y_t - \hat{y}_{ijt}||^2 \tag{1}$$

We identify the position ij of the ground-truth y_0 object at time $t = 0$ (the current time-step). Then we extract the prediction $\hat{y}_{ij} = [\hat{y}_{ij1}, ..., \hat{y}_{ijT}]$ that corresponds to that grid-cell and calculate the l2 loss starting at $t = 1$ up to the end of the prediction at $t = T$.

Unfortunately, we cannot use this architecture and loss if the tracking id's are unknown. We will propose adaptations to the neural network architecture and loss. Unlike previous work, we do not need a tracking id during training and inference. We visualize the architecture in figure 3. The main difference to the tracking-based case in figure 2 is an additional prediction step at $t = 1$. The convolutional neural network predicts the likely location of detections at $t = 1$ using a Gaussian Mixture with modes arranged on a 128 x 128 grid (like anchors in object detection). Each grid cell with indices m, n corresponds to a 0.5 m x 0.5 m area. If an object is likely to be detected in one of the grid cells, the Gaussian Mixture mode is active $\phi_{mn} > 0$.

On top of the prediction at time-step $t = 1$, the convolutional neural network predicts how the mean of each mode of the Gaussian Mixture model propagates through time. Thereby, we obtain object identities that are not explicitly labelled in the training data-set, which only contains raw 3D object detections or unreliable tracking id's. Note that we do not need to know the number of objects in advance, some detections up to $t = 0$ may be missing, and new objects may appear later.

First, we predict the position of all objects at $t = 1$ using a Gaussian Mixture. Why do we not do this at $t = 0$? Because if there was a false negative, then we do not have a ground truth that would tell us that there is a false negative. $t = 0$ is part of the input, i.e., the neural network sees if it exists or not, while $t = 1$ is part of the prediction, i.e., the neural network does not get to see it even if it exists. Figure 4 illustrates the distinction further.

Since we have no associations (no tracking id's), we need a loss function that deals with all ground-truth objects (i.e., noisy detections) and every prediction (Gaussian modes) at the same time (see figure 5). The following loss function does that (log-likelihood loss).

Fig. 5. We consider the predictions at a future time-step. On the left, we have tracking information and can associate the ground truth (yellow) and the prediction (blue). One prediction matches one ground truth. The remaining ground-truth objects (grey) are irrelevant as their identities differ. On the right, we have multiple ground-truth objects and multiple predictions (significant Gaussian Mixture modes). The loss function should not assume an association as we do not have any tracking information but include all ground-truth objects and predictions at the same time.

$$L = -\frac{1}{T}\sum_{t=1}^{T}\log\left(\frac{1}{I^2}\sum_{m,n=1}^{I}\phi_{mn}\mathcal{N}_{mn}(y_t|\hat{\mu}_{mnt};\sigma=1)\right) \tag{2}$$

We do not propagate the gradient into the Gaussian mode weights ϕ_{mn} for $t > 1$. We use only the first time-step $t = 1$ to predict objects. There are two reasons for that: (1) Training is much more stable, and the prediction performance is much better (2) We want to reason about objects that exist close to $t = 0$, not objects that might appear 6 seconds later that we have not detected yet.

Final Displacement Error

We evaluate the long-term prediction performance with the final displacement error.

$$FDE = ||y_T - \hat{y}_T|| \tag{3}$$

y_T is the ground-truth position at the last time-step and $\hat{y}_T$ is the predicted position at the last time-step.

It is not straightforward to compare the tracking based architecture (figure 2 and loss (1)) and detection based architecture (figure 3 and loss (2)). For the tracking-based case, a tracking algorithm tells us where an object is at $t = 0$. Then we predict the motion T time-steps into the future. Without the tracking component, we first need to identify objects at $t = 1$ (from the ground-truth) and then predict their motions $T-1$ time-steps into the future, i.e., we evaluate over a prediction horizon of $t = [2, ..., T]$ for both methods. One approach takes the object identity at $t = 0$ and the other at $t = 1$. To compensate the advantage (of the method without tracking id), we subtract the tracking-based approach's final displacement error for $t = 1$.

IV. EXPERIMENTS

In this section we answer five questions:

1) How do predictions change when we do not use the tracking id in the loss function?
2) How do the predictions change when we drop a significant amount of labels?
3) How do predictions change if the 3D-positions have a positional uncertainty?
4) How do predictions change if we assume realistic detector noise?
5) How much additional data would we need to compensate for any performance decrease in 2), 3) and 4)?

In 1) we consider the architecture changes in figures 2, 3 and loss functions (1), (2) for a model that uses a tracking id in the loss vs one that does not.

In 2) we simulate detector noise by removing 20% of the ground-truth labels at random (uniform). Since nuScenes has a sampling rate of 2 Hz, missing labels can drastically affect the performance compared to, e.g., 10 Hz, where it is much easier to interpolate missing information. Dropping a label in the middle of a sequence corresponds to a full second without any detections.

In 3) we add Gaussian noise with a standard deviation of 0.5 m to the ground-truth locations. This simulates the positional noise of a 3D object detector. Since we consider at most three data points in the input (at 2 Hz), the neural network cannot easily recover the ground-truth locations.

In 4) we infer reasonable noise values from the literature. Namely, 15 % of labels missing and a positional uncertainty of 0.3 m.

In 5) we reduce the data-set size by excluding a certain number of scenes – 20-second sequences–in the training data. We evaluate the performance on 30 % 50%, 57.5 %, 65 %, 80% and 100% of the data and see how 2), 3) and 4) compare.

NuScenes Dataset

We evaluate our method on the large-scale public data-set nuScenes [29]. It contains 1000 selected sequences from Boston (right-hand driving) and Singapore (left-hand driving), each 20 seconds long. We use the train/validation split of the official nuScenes prediction challenge, which divides the data into sequences with one second of history and six seconds of future at a sampling rate of 2 Hz. Note that the official leaderboard evaluates any method on the validation set. We create rasterized images using the official nuScenes devkit that include objects within a 120-meter radius around the data-capturing vehicle (at the "current" time step between history and future) (see figure 1). The image is centred at the position of the data-capturing vehicle and oriented into the driving direction. We will not consider the data-capturing vehicle in the prediction task but all other vehicles (vulnerable road users are not part of the validation set). We include the data-capturing vehicle in the input in case its behaviour influences another driver that we want to predict.

Hyper-parameters

The Unet takes an input feature size of 60, with 0.01 weight decay and a batch size of 20 for all experiments. We use AdamW [30] for optimizing the loss function. The learning rate for the tracking-based architecture and loss is 0.001, with a subsequent drop to 0.0001 after the final displacement error does not improve on the validation set for 30 epochs. We wait for at least another 30 epochs of no improvement before we finish training. The learning rate for the architecture and

TABLE I

FINAL DISPLACEMENT ERROR (METERS) OF OTHER METHODS BASED ON CONVOLUTIONAL NEURAL NETWORKS ON THE PUBLIC NUSCENES BENCHMARK. ALL METHODS INCLUDING OURS (FIGURE 2 AND LOSS (1)) USE TRACKING ID'S IN THE LOSS FUNCTION.

Time	ours	P2T[8]	Multipath[7]	Covernet[6]	MTP[3]
6 s	10.4	10.5	10.4	9.3	9.2

TABLE II

FINAL DISPLACEMENT ERROR (METERS) WHEN TRAINING ON NOISY LABELS (B, C, D) AND PERFECT LABELS (A, AND OURS). THE CAPITAL LETTERS CORRESPOND TO THE SUBSECTIONS IN THE EXPERIMENTAL SECTION. 'OURS' REFERS TO THE SAME METHOD AS IN TABLE I.

Time	A	B	C	D	ours (with track id)
5.5 s	10.1	11.1	10.9	11.6	10.1

loss without a tracking id is 0.0001 for the whole time. We waited for at least 30 epochs where the final displacement error did not improve on the validation set before finishing the training.

Comparison to Benchmark

Before exploring the effects of 3D object detector noise, we compare our tracking-based algorithm with other convolutional neural network based prediction methods on the nuScenes benchmark. Table I shows the other recently published methods next to ours. We want to stress that we investigate a new setup with noisy labels. It is not our primary goal to outperform state-of-the-art methods, but it is essential to show that we experiment with a method comparable to others.

A. No Tracking ID in Loss

We ran the model with and without tracking information. We did not find a meaningful difference in performance with both approaches at around 10.1 m final displacement error (see table II). The tracking id itself may not be relevant to the performance of the prediction algorithm if the 3D object detections are free of noise. [21] also found a weak dependence of the prediction performance in the input (loss is tracking-based). The winning team of the Lyft challenge on Kaggle also remarked[1] that their convolutional neural network did not show a meaningful difference in performance when switching tracking id's off in the input (loss tracking-based). As our experimental result indicates, the same may be true for the loss function as well.

B. Removing Labels

We want to understand how noisy 3D object detections from an on-board perception system affect our prediction algorithm. In our first experiment, we remove 20 % of the labels at random in the training set, i.e., the label information is permanently lost. The validation set remains untouched. This corresponds to a detection recall of 80 %.

While the tracking id appears to be of minor importance, a loss of 3D labels does decrease the prediction performance from 10.1 m to 11.1 m final displacement error – a 10% decrease (see table II).

[1]https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/discussion/201493 (as of January 31st 2021)

C. Adding Positional Noise

A 3D object detector provides noisy estimates of the position of a vehicle. We add Gaussian noise with a standard deviation of 0.5 m around the ground-truth positions in the training set to simulate this effect. The validation set remains untouched. We assume the typical vehicle bounding box to be 3.5 m x 6 m (as suggested by [14]), and we only consider detections with an intersection over union of at least 0.7 as true positives. Then 0.5 m positional noise corresponds to a precision of 56%, i.e., 44% of detections end up as false positives. Note that the convolutional neural network receives only three labels for the current time-step and the past, making it hard to filter the noise. We find that the prediction performance degrades from 10.1 m to 10.9 m final displacement error – an 8% decrease (see table II).

D. Realistic Detector Noise

A typical 3D object detector would achieve a recall of at least 83 % and a precision of at least 90% at an intersection over union of at least 0.7 [14]. We simulate these performance metrics by removing 15% of the labels and adding a positional noise of 0.3 m (on training set, not validation set). These two sources of noise result in a recall of 83% and a precision of 88 % given an intersection over union of at least 0.7 and a typical vehicle bounding box of 3.5 m x 6 m. We find that the prediction performance degrades from 10.1 m to 11.6 m final displacement error – a 16% decrease (see table II).

E. How Much Data Do We Need?

The original motivation for detection based prediction is the large amount of low band-with data that a fleet of millions of cars can provide. So how much data do we need to compensate for a performance decrease of 16 % (realistic detector noise)? To get an intuition, we train the same algorithm with the same hyper-parameters[2] on 30 % - 100 % of the data (we remove scenes – 20-second sequences – at random). We visualize the results in figure 6 and compare them to the noise experiments in the previous subsections. As we can see, 100 % of noisy training data results in a similar motion prediction performance as 50 % - 70 % of high-fidelity labels (depending on the experiment). That is remarkable given the data efficiency of 3D object detections to raw sensor data. Not to mention the additional effort of labelling the sensor data.

[2]A smaller data set leads to over-fitting. We would need to adjust the hyper-parameters such as network capacity, batch size and learning rate. However, we believe the differences would not change the interpretation.

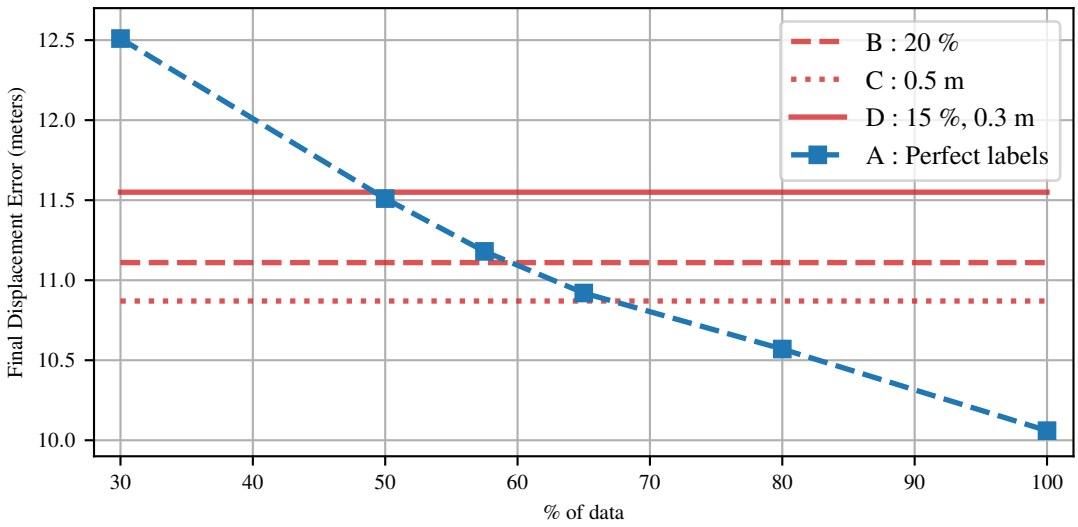

Fig. 6. Red: We evaluate B, C and D on **100 %** of the data. In B, 20 % of 3D detections are missing, and in C, all 3D detections have a positional uncertainty of 0.5 m. In D, we have: 15 % of 3D detections are missing, and all the remaining detections have a positional uncertainty of 0.3 m. D is of particular interest as this is similar to the performance of a real-life 3D object detector. Blue A: prediction model without tracking id's trained on 30%-100 % of the training data. The intersection of the blue and red graphs is between 50%-70 %. **Implications**: We can get the same result on 100 % of noisy 3D detections as with at least 50 % of high-fidelity labels. The former needs only an on-board perception system and transfers efficient 3D labels. The latter needs separate manual or automated labelling pipelines, and we have to transfer inefficient raw sensor data from data-gathering vehicles.

V. CONCLUSIONS

We discussed how to design a human motion prediction algorithm to train on noisy data (missing detections, positional uncertainty, no tracking id's or unreliable tracking id's) from an on-board perception system. First, we explored the difference between a method that uses tracking id's and our proposed method that does not use any tracking id's in the training data. We did not find a significant difference. Detector noise affects our training data, and we analyse the performance degradation. Missing labels (removing up to 20% of labels at random) and positional uncertainty (adding up to 0.5 meters Gaussian noise) affected the performance. However, we can show that increasing the training set size by a factor of two should compensate for that degradation in performance (100 % of noisy labels equals 50 % of high-fidelity labels). While we need to increase the training set size to compensate for the noise introduced by an on-board perception system, that increase is multiple orders of magnitude smaller than the amount of low-bandwidth data (3D-detections and map locations) that we can collect from a large fleet of vehicles.

In future work, we would like to expand the analysis to multi-modal motion prediction algorithms and investigate methods other than convolutional neural networks such as ones based on PointNets/Voxels, e.g., [13] that look promising.

VI. ACKNOWLEDGEMENTS

This project has received funding from the German Federal Ministry of Economics and Energy under the @City project (grant ID: 19 A 17015 A).

REFERENCES

[1] K. T. e. a. H. Caesar, J. Kabzan, "Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," in *CVPR ADP3 workshop*, 2021.

[2] L. A. De La Fuente Layos, "Passenger mobility in Europe," *Eurostat, ISSN 1977-0324, Catalogue number KS-SF-07-087-DE-C*, 2007.

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

[4] J. Hong, B. Sapp, and J. Philbin, "Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8454–8462.

[5] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, N. Singh, and J. Schneider, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2095–2104.

[6] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "CoverNet: Multimodal Behavior Prediction Using Trajectory Sets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction," ser. Proceedings of Machine Learning Research, vol. 100, 2020, pp. 86–99.

[8] N. Deo and M. M. Trivedi, "Trajectory Forecasts in Unknown Environments Conditioned on Grid-Based Plans," *CoRR*, vol. abs/2001.0, 2020.

[9] N. Djuric, H. Cui, Z. Su, S. Wu, H. Wang, F.-C. Chou, L. S. Martin, S. Feng, R. Hu, Y. Xu, A. Dayan, S. Zhang, B. C. Becker, G. P. Meyer, C. Vallespi-Gonzalez, and C. K. Wellington, "MultiXNet: Multiclass Multistage Multimodal Motion Prediction," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 435–442.

[10] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, and B. Sapp, "MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction," 2021.

[11] X. Weng, Y. Yuan, and K. Kitani, "PTP: Parallelized Tracking and Prediction with Graph Neural Networks and Diversity Sampling," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4640–4647, 2021.
[12] J. Gu, Q. Sun, and H. Zhao, "DenseTNT: Waymo Open Dataset Motion Prediction Challenge 1st Place Solution," *arXiv preprint arXiv:2106.14160*, 2021. [Online]. Available: http://arxiv.org/abs/2106.14160
[13] M. Ye, T. Cao, and Q. Chen, "TPCN: Temporal Point Cloud Networks for Motion Forecasting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 318–11 327.
[14] 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 Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.
[15] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila, "Crafted vs Learned Representations in Predictive Models—A Case Study on Cyclist Path Prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 4, pp. 747–759, 2021.
[16] J. F. P. Kooij, N. Schneider, F. Flohr, and D. M. Gavrila, "Context-based pedestrian path prediction," in *European Conference on Computer Vision*. Springer, 2014, pp. 618–633.
[17] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: a survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.
[18] D. Nuss, S. Reuter, M. Thom, T. Yuan, G. Krehl, M. Maile, A. Gern, and K. Dietmayer, "A random finite set approach for dynamic occupancy grid maps with real-time application," *The International Journal of Robotics Research*, vol. 37, no. 8, pp. 841–866, 2018.
[19] P. Wu, S. Chen, and D. N. Metaxas, "MotionNet: Joint Perception and Motion Prediction for Autonomous Driving Based on Bird's Eye View Maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 385–11 395.
[20] X. Weng, J. Wang, S. Levine, K. Kitani, and N. Rhinehart, "4D Forecasting: Sequantial Forecasting of 100,000 Points," in *Euro. Conf. Comput. Vis. Worksh*, vol. 3, 2020.
[21] A. Trabelsi, R. J. Beveridge, and N. Blanchard, "Drowned out by the noise: Evidence for Tracking-free Motion Prediction," pp. 1–12, 2021. [Online]. Available: http://arxiv.org/abs/2104.08368
[22] S. Krebs, M. Braun, and D. M. Gavrila, "Generating 3D Person Trajectories from Sparse Image Annotations in an Intelligent Vehicles Setting," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 783–788.
[23] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3D Object Detection from Point Cloud Sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6134–6144.
[24] Q. Meng, W. Wang, T. Zhou, J. Shen, Y. Jia, and L. Van Gool, "Towards A Weakly Supervised Framework for 3D Point Cloud Object Detection and Annotation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
[25] H. Tian, Y. Chen, J. Dai, Z. Zhang, and X. Zhu, "Unsupervised Object Detection With LIDAR Clues," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2021, pp. 5962–5972.
[26] 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 Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.
[27] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 683–700.
[28] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
[29] 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," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2020.
[30] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101