electronics

MDPI

Article

Recursive Least Squares Based Refinement Network for Vehicle Trajectory Prediction

Shengyi Li [1,†], Qifan Xue [1,†], Dongfeng Shi [2], Xuanpeng Li [1,*] and Weigong Zhang [1]

[1] School of Instrument Science and Engineering, Southeast University, Nanjing 211189, China; li_shengyi@seu.edu.cn (S.L.); xue_qifan@seu.edu.cn (Q.X.); zhang_weigong@seu.edu.cn (W.Z.)
[2] Beijing Remote Sensing Information Institute, Beijing 100192, China; xinghelion@163.com
* Correspondence: li_xuanpeng@seu.edu.cn
† These authors contributed equally to this work.

Abstract: Trajectory prediction of surrounding objects plays a pivotal role in the field of autonomous driving vehicles. In the current rollout process, it suffers from an accumulation of errors, which has a negative impact on prediction accuracy. This paper proposes a parametric-learning recursive least-squares (RLS) method integrated with an encoder–decoder framework for trajectory prediction, named the recursive least-squares-based refinement network (RRN). Through the generation of several anchors in the future trajectory, RRN can capture both local and global motion patterns. We conducted experiments on the prevalent NGSIM and INTERACTION datasets, which contain various scenarios such as highways, intersections and roundabouts. The promising results indicate that RRN could improve the performance of the rollout trajectory prediction effectively.

Keywords: recursive refinement network; trajectory prediction; parametric-learning recursive least square; anchor generator

Citation: Li, S.; Xue, Q.; Shi, D.; Li, X.; Zhang, W. Recursive Least Squares Based Refinement Network for Vehicle Trajectory Prediction. *Electronics* **2022**, *11*, 1859. https://doi.org/10.3390/electronics11121859

Academic Editor: Jose Eugenio Naranjo

Received: 20 May 2022
Accepted: 9 June 2022
Published: 12 June 2022

1. Introduction

Trajectory prediction is a fundamental function of autonomous driving vehicles in traffic conditions. Autonomous vehicles must make decisions carefully based on trajectories of surrounding objects, e.g., various vehicles, cyclists and pedestrians in order to reduce the risk of collisions [1]. Trajectory prediction becomes an indispensable work for both perception and planning in autonomous driving systems [2,3]. History trajectories of ego and neighboring agents are utilized to predict their future position, which helps plan plausible paths for traffic agents. The challenge lies in the tendency of the trajectories to be of high uncertainty due to unobserved intent and complex interactions between multiple agents.

A number of studies have been conducted to predict trajectories from traditional machine learning and state-of-the-art deep neural networks. Recent works suggest employing Recurrent Neural Network (RNN) [4] to predict trajectories. Additionally, the social mechanism and the attention mechanism [5] are used to take the interaction of agents into account. Moreover, variational auto-encoder (VAE) [6] and generative adversarial networks (GAN) [7] are employed to improve the generalization of models. Some studies also consider maps [8] and traffic rules [9] as important context information. These studies adopt the same logic, which forecast the future positions in a rollout way. The term 'rollout' refers to the sequence generation process, where the prediction at each step is put into the model at the next time step [10]. However, there are some limitations in most studies up to now.

One limitation is that the rollout mechanism leads to error accumulation during the recursive process. It attempts to optimize the performance of prediction of several states (position, velocity, etc.) at each step. Errors accompanied by the state's update have negative influence on the next prediction. Thus, prediction by the rollout mechanism

becomes less and less accurate over time. In addition, the rollout mechanism fails to capture the global motion feature, since these algorithms are designed to run step by step based on the history trajectories of multiple agents.

In the topic of trajectory prediction, local motion pattern is related with the motivation of drivers to perform certain driving behaviors within short range, such as immediate change of orientation to avoid collision, sudden acceleration to overtake, and so on. The global motion pattern is regarded as the target of driving to perform long-term driving behaviors. Local motion patterns change frequently in a short period of time, while the global motion pattern changes less over time and remains stable over a longer period of time. Although some work similar to long short-term memory (LSTM) [11] proposes a balance structure to form a trade-off between local and global motion patterns, the short-term or local motion remains predominant due to the simple logic of the forgotten gate. It is still difficult to capture long-term or global motion patterns in the rollout prediction methods.

In addition, many researchers tend to use context information such as maps [8] or traffic rules [9] into their models. Context information provides more constraints on trajectory prediction, but it is not clear whether it makes an impact on the generalization of prediction model [12–14]. As a result, context-free approaches are also of great importance for trajectory prediction, which focuses on the interactive motion of multiple agents.

This paper proposes a context-free model named the recursive least-squares-based refinement network (RRN) to improve rollout models by integrating recursive least squares (RLS) [15] in a parametric learning way. RLS is a well-known adaptive filter algorithm that efficiently updates a weighted linear least-squares cost as new data become available. In order to improve the ability to model global motion pattern, we introduce 'anchors' to capture global-consistent features in the rollout process. As shown in Figure 1, the anchors are defined as the possible positions in the future trajectory at certain time steps. At each time step, the refinement module leverages the predicted anchors to improve the rollout estimation based on RLS.

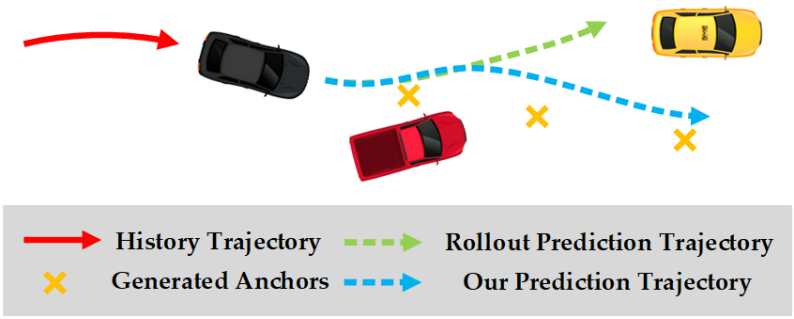

Figure 1. Overview of our work. The red line represents the history trajectory of certain object. The yellow crosses denote the generated anchors. The dotted green and blue lines represent trajectories predicted by the rollout method and our proposed RRN method, respectively.

Experiments are conducted on the NGSIM [16] and INTERACTION [17] datasets. They are widely used datasets in the field of trajectory prediction [13,14,18,19]. Experiments verify the promising performance of our method both qualitatively and quantitatively.

Our contributions are summarized as follows:

1. We propose an RLS-based framework with generated anchors that captures both local and global motion patterns of surrounding vehicles. This method manages to minimize the negative impact of error accumulation.
2. Instead of artificial parameters, we propose a data-driven RLS method that allows the covariance matrix of RLS to be automatically updated by a learning-based method. Our prediction network based on the proposed RLS model can be more accurate and robust in various traffic scenes.

3. Our model can be easily embedded into most rollout approaches. An ablation study has proved the effectiveness of our module which can be easily plugged into various networks to improve their performance.

This paper is organized as follows: In Section 2, recent work on trajectory prediction is summarized. In Section 3, our method is presented including the anchor generator, network framework and parametric-learning recursive least-squares model. In Section 4, our method is evaluated on real-world datasets and analyzed both qualitatively and quantitatively. Finally, the conclusion is presented in Section 5.

2. Related Works

Trajectory prediction can be classified into three types of methods, i.e., physics-based methods, pattern-based methods, and planning-based methods [20]. Physics-based methods estimate the motion state of the agent using explicit dynamical models based on Newton's law of motion. Pattern-based methods learn prototypical trajectories from observed agent behavior to predict future motion. Planning-based methods include reasoning about the possible goals and corresponding paths based on optimization method. Among these methods, pattern-based methods benefit from the thriving development of deep learning methods, which can capture a powerful representation about motion patterns from observed trajectories of agents.

2.1. Deep Learning in Trajectory Prediction

The introduction of generative models provide the prediction with multi-modality, which means the model will finally output a number of predicted trajectories with corresponding confidence. Lee et al. [21] propose a Conditional Variational Autoencoder (CVAE), named DESIRE, to generate multiple future trajectories based on agents' interaction, scene semantics and expected reward function. Tang et al. [22] introduce a probabilistic framework called Multiple Future Prediction (MFP) that efficiently learns latent variables to jointly model the multi-step future motions between multiple agents.

Numerous studies have also attempted to model social interaction. Alahi et al. [23] propose a Social LSTM model which introduces a social-pooling mechanism to aggregate the neighboring agents' behaviors together. Deo et al. [24] employ the convolution operation to improve the performance of the social pooling method. GAN [25] is integrated with Social LSTM for trajectory prediction. These models succeed in modeling social interaction in a parametric learning way, causing a significant improvement in prediction accuracy. This study does not focus on vehicle interaction, instead we attempt to solve problems caused by the rollout mechanism.

2.2. Seq2seq Trajectory Prediction

In most deep learning frameworks, sequence-to-sequence (seq2seq) models are commonly adopted by researchers to build the network architecture of sequence prediction [26]. Seq2seq originates from neural language processing, which is a general-purpose encoder–decoder framework. RNNs and their variants, e.g., LSTM [11], and gated recurrent unit (GRU) [27] are classical sequence-to-sequence models under the rollout frameworks. Recently, they have also been applied in trajectory prediction [21,28–32]. In [28], LSTM is used to track the position of the object based on the ranging sensor measurements. In [29], the driver's intention is identified based on the trajectory data using LSTM. In [30], LSTM is applied to predict the location of the vehicle after a certain number of seconds using past trajectory data. In [21], GRU is combined with conditional variational auto-encoder (CVAE) to predict the vehicle trajectory. Seong et al. [31] propose a similar structure which produces the K most likely trajectory candidates from the decoder over occupancy grid map by employing the beam search technique. It achieves significantly higher accuracy over the conventional trajectory prediction techniques. Du et al. [32] further improve the performance of trajectory prediction based on the encoder–decoder architecture via fusion of attention mechanism. Seq2seq models are proved to be more powerful than traditional

methods when dealing with temporal prediction task. However, it generates the predicted value in a rollout way, and the weakness of the rollout method is not accounted for by most researchers. They also lack the ability to capture global motion patterns even if LSTM is designed to balance the long-term and short-term memory. In order to solve these problems, some works attempt to add extra information such as goal point and intent, which are regarded as long-term features from a global perspective.

2.3. Goal-Conditioned Methods

Many researchers have attempted to make an estimation of goals [33–38], intents [39–41] and anchors [42,43] to benefit the trajectory prediction.

Some researh works have tried to set 'goal points' or 'end points' to reduce accumulative errors in the rollout approaches. TPNet [33] makes final predictions by refining the proposal trajectories, which are generated by regressed endpoints and polynomial fitting. PECNet [34] infers distant endpoints of trajectories to assist in long-range multi-modal trajectory prediction, together with the social pooling mechanism. Goal-GAN [35] presents a GAN-based end-to-end trainable model for human trajectory prediction with the goal estimation module. Zhao [36] proposes to predict the trajectories of agents via the guidance of goal expertise, which can be obtained with modest expense through a novel goal-search mechanism on already-seen training examples. TNT [37,38] performs target-conditioned motion estimation for trajectory prediction and combines expert knowledge (e.g., HD maps) to make constraints on traffic scenes.

Intent prediction represents another possible solution to improve the performance of trajectory prediction. IntentNet [39] manually defines several common motion categories for self-driving vehicles, such as left turn and lane changes, and learned a separate motion predictor for each intent. This manual categorization depends on task and dataset, which may be too coarse to capture intra-category multimodality. In [40], the model generates multi-modal trajectory possibility prediction with high interpretability according to the estimation of driver intention. Li et al. [41] propose a conditional deep generative model that combines a graph attention network built upon inter-agent latent code. These methods have inspired us to introduce a group of stochastic variables regarded as 'anchors' to reduce the accumulative errors. Anchors are initialized used in the field of object detection [44]. More recently, MultiPath [42] and CoverNet [43] chose to quantize the trajectories into anchors, where the trajectory prediction task is reformulated into anchor selection and offset regression. The anchors are either pre-clustered into a fixed priori set or obtained dynamically based on kinematic heuristics. Here, we propose a novel anchor generation strategy in the RLS framework to make predictions on the future trajectories of multiple vehicles.

3. Recursive Least-Squares Based Refinement Network

3.1. Problem Definition

In this work, vehicle trajectory prediction is formulated as estimating the conditional probability distribution $p(Y|X)$ of future positions $Y_i = \{(x_i^t, y_i^t) \in \mathbb{R}^2 | t = t_{obs} + 1, \cdots, t_{pred}\}$ based on the observed trajectories $X_i = \{(x_i^t, y_i^t) \in \mathbb{R}^2 | t = 1, 2, \cdots, t_{obs}\}$ of N currently visible vehicles. We assume that the conditional probability distribution $p(Y|X)$ follows the bivariate Gaussian distribution as

$$Y_i \sim \mathcal{N}(\mu_i, \Sigma_i), \tag{1}$$

$$\mu_i = \begin{pmatrix} \mu_{x,i} \\ \mu_{y,i} \end{pmatrix}, \Sigma_i = \begin{pmatrix} \sigma_{x,i}^2 & \sigma_{x,i}\sigma_{y,i}\rho_i \\ \sigma_{x,i}\sigma_{y,i}\rho_i & \sigma_{y,i}^2 \end{pmatrix}, \tag{2}$$

where μ_i and Σ_i denote the mean and covariance of predicted positions of vehicle i in $x - y$ axes.

In the trajectory prediction, a set of anchors $A = \{A_i | i = 1, 2, \cdots, N\}$ are proposed to facilitate the estimation of the probability distribution $p(Y|X)$ as

$$p(Y|X) = \sum_{A} p(Y, A|X) = \sum_{A} p(Y|A, X)p(A|X), \tag{3}$$

where $A_i = \{(x_i^t, y_i^t) \in \mathbb{R}^2 | t = t_{obs} + k\}$ and $k \in \{1, 2, \cdots, \tau | \tau < t_{pred} - t_{obs}\}$ denotes the time step when anchors are generated and inserted.

The first component can be further factorized as

$$p(Y|A, X) = \prod_{t=t_{obs}+1}^{t_{pred}} p(Y^t|Y^{t-1}, A^{t-1}, X)\Omega^t + p(Y^t|Y^{t-1}, X)(1 - \Omega^t), \tag{4}$$

$$\Omega^t = \begin{cases} 1 & \text{anchor available at time } t \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

The second component $p(A|X)$ represents the latent motion patterns of agents based on the anchor generation in certain time steps. Anchor generator provides the global motion patterns, which is not considered in most rollout methods [11,27,45]. We attempt to manage the problem of imbalance between local and global motion patterns by using an anchor generator.

3.2. Anchor Generator

In the anchor generator, the bivariate Gaussian distribution is used to characterize the conditional probability distribution $P(A|X)$ as

$$A_i \sim \mathcal{N}(\mu_i, \Sigma_i), \tag{6}$$

and $\hat{Z} = \{\mu_i | i = 1, 2, \cdots, N\}$ and $\hat{R} = \{\Sigma_i | i = 1, 2, \cdots, N\}$ denote the mean and covariance of all anchors. As shown in Figure 2, anchors are then inserted into the seq2seq framework to adjust accumulated errors in the rollout process. These anchors are generated only once in the process of trajectory prediction. They are completely conditionally independent each other and can provide global information.

The anchor generator is composed of three parts: an input embedding layer, a multilayer perceptron (MLP) and an output layer. Input embedding layer is a linear layer mapping of the dimension of input X from 2 to $C_{emb} = 64$. MLP is composed of two linear layers, which generate features with the dimension $C_{mlp} = 128, 64$. The output layer generates the predicted anchor which consists of 5 variables, i.e., μ_x, μ_y, σ_x, σ_y, and ρ. We use the Rectified Linear Unit (ReLU) as the activation function. These three parts are conducted as follows:

$$h_{emb} = Linear(X; W_{emb}), \ \ h_{emb} \in \mathbb{R}^{N \times \tau \times C_{emb}}, \tag{7}$$

$$h_{mlp} = MLP(ReLU(h_{emb}); W_{mlp}), \ \ h_{mlp} \in \mathbb{R}^{N \times \tau \times C_{mlp}}, \tag{8}$$

$$A_{out} = Linear(ReLU(h_{mlp}); W_{out}), \ \ A_{out} \in \mathbb{R}^{N \times \tau \times 5}, \tag{9}$$

where h_{emb} and h_{mlp} denote the features generated by input embedding layer and MLP. $\hat{Z}$ and $\hat{R}$ are further calculated based on A_{out}, which represent the final results by the anchor generator. W_{emb} denotes the weights of the input embedding layer, W_{mlp} refers to the weights of the MLP, and W_{out} denotes the weights of the output layer. Generated anchors are further used by the RLS module to adjust the accumulative errors from a global perspective as shown in Figure 2.

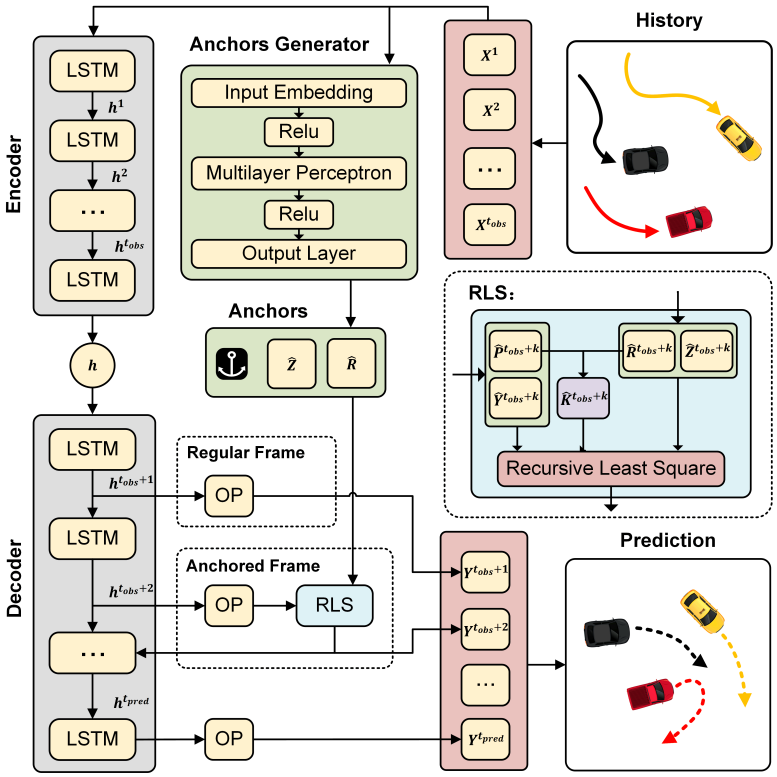

Figure 2. The pipeline of RRN. X denotes the history trajectories of vehicles. $\hat{Y}$ and $\hat{P}$ denote the mean and covariance characterizing the probability distribution over predicted trajectories. $\hat{Z}$ and $\hat{R}$ denote the mean and covariance characterizing the probability distribution over anchors. K denotes the gain matrix of RLS module. 'OP' denotes the output layer of decoder. Trajectory encoder and decoder are both implemented by a couple of LSTM units. A bunch of anchors are generated from a global perspective. At several time steps, the rollout estimation is refined by RLS using information provided by anchors. 'Regular frame' refers to the frame without anchor adjustment, while 'Anchored frame' refers to the one with anchors.

3.3. Trajectory Encoder

The sequence-to-sequence structure is a general framework for sequential prediction problems. The structure consists of an encoder and a decoder. We employ LSTM as an encoder, which recurrently takes each step in the sequence as input and updates the state of LSTM unit [11].

As shown in Figure 2, the trajectory encoder is composed of a series of LSTM units via sharing weights. The encoder takes history trajectories $X \in \mathbb{R}^{N \times t_{obs} \times 2}$ as input. The input embedding layer also maps the dimension of coordinates from 2 to $C'_{emb} = 32$ as mentioned in Section 3.2.

At each time step, LSTM receives the current embedded tensor as input and update its hidden state together as shown in Figure 2. In the recurrent process, feature dimension is further mapped from $C'_{emb} = 32$ to $C_{enc} = 64$ within LSTM module. The process can be presented as follows:

$$h_{emb} = Linear(X; W_{emb}), \quad h_{emb} \in \mathbb{R}^{N \times t_{obs} \times C'_{emb}}, \tag{10}$$

$$h^t_{enc} = LSTM_{enc}(h^{t-1}_{enc}, h^{t-1}_{emb}; W_{enc}), \quad t = 1, \cdots, t_{obs}, \quad h^t_{enc} \in \mathbb{R}^{N \times C_{enc}}, \tag{11}$$

where h_{emb} denotes the embedded features. W_{emb} denotes the weights of the input embedding layer in the encoder part, and W_{enc} refers to all weights of LSTM module. Finally, the encoder generates a hidden feature h^t_{enc} as input of the trajectory decoder.

3.4. Trajectory Decoder

Trajectory encoder takes history trajectories as input and estimates the hidden motion state of vehicles. Then, the decoder is responsible for generating future trajectory based on the hidden status. The structure of the trajectory decoder is composed of δ LSTM units ($\delta = t_{pred} - t_{obs}$) with sharing weights as shown in Figure 2. The decoder process can be formulated as follows:

$$h_{dec}^{t} = LSTM_{dec}(h_{enc}^{t-1}, h_{dec}^{t-1}; W_{dec}), \ \ t = t_{obs} + 1, \cdots, t_{pred}, \ \ h_{dec}^{t} \in \mathbb{R}^{N \times (t_{pred} - t_{obs}) \times C_{dec}}, \quad (12)$$

$$h_{out} = Linear(h_{dec}^{t}; W_{out}), \ \ h_{out} \in \mathbb{R}^{N \times (t_{pred} - t_{obs}) \times 5}, \quad (13)$$

where W_{dec} refers to weight matrices of all LSTM units and W_{out} denotes the weights of output layer. C_{dec} refers to the feature dimension of the decoder as 32. We can further get the mean vectors $\hat{Y}$ and covariance matrices $\hat{P}$ based on $h_{out} = \{\mu_x, \mu_y, \sigma_x, \sigma_y, \rho\}_{1:N}^{t_{obs}+1:t_{pred}}$, which characterizes the probability distribution of all predicted trajectories.

3.5. Recursive Least-Squares Module

In our method, the recursive least-squares module is combined with a trajectory decoder part in order to adjust the accumulative errors and capture the global motion features of vehicles. Anchors are not simply concatenated to the decoder LSTM unit as observed in many 'Goal points' methods, but a parametric-learning RLS module is proposed to produce dynamic filtering with generated anchors at certain time steps.

RLS is a recursive method of aggregating two or more random variables to minimize the variance [46]. In traditional RLS, some parameters such as noise covariance matrix should be set artificially by hand. Machine learning methods open the window to set these parameters automatically. To date, there are some studies which have investigated the RLS in deep learning methods, but none in trajectory prediction. RLS provides a novel view on the fusion of goal points and the rollout prediction. Our purpose is to employ anchors $\hat{Z}$ and $\hat{R}$ and rollout predictions $\hat{Y}$ and $\hat{P}$ in the RLS to generate a refined estimation for future coordinates.

At the time step when anchors are generated, RLS module takes $(\hat{Z}^t, \hat{R}^t)$ and $(\hat{Y}^t, \hat{P}^t)$ as input. RLS module sequentially computes the gain and refines rollout predictions as

$$K^t = \hat{P}^{t-1} H^{t^T} (H^t \hat{P}^{t-1} H^{t^T} + \hat{R}^t)^{-1}, \quad (14)$$

$$\hat{Y}^t = \hat{Y}^{t-1} + K^t (\hat{Z}^t - H^t \hat{Y}^{t-1}), \quad (15)$$

$$\hat{P}^t = (I - K^t H^t) \hat{P}^{t-1}, \quad (16)$$

where H denotes the vector $[1, 1]^T$, K denotes the gain, and I is set to the identity matrix here. The refined estimations by the RLS module are then fed back into the next LSTM unit, as shown in Figure 2.

Finally, we use negative log-likelihood (NLL) loss to be the loss function as

$$\text{NLL} = -\sum_{t=t_{obs}+1}^{t_{pred}} log(\mathcal{N}(\hat{Y}^t, \hat{P}^t)). \quad (17)$$

The output of trajectory decoder is refined by the parametric-learning RLS module, which can make a tradeoff between local and global motion patterns. Consequently, this framework is able to reduce accumulated errors.

4. Experiments

In this section, the performance of our method is evaluated on the public NGSIM [16] and INTERACTION [17] datasets. NGSIM are a set of video-transcribed data of vehicle trajectories on US-101, Lankershim Blvd. in Los Angeles, and I-80 in Emeryville. In total, it

contains approximately 45 min of vehicle trajectory data collected at 10 Hz. INTERACTION is a large-scale real-world dataset which consists of top-down scenes from intersections, highways and roundabouts. The data are collected from three different continents (North America, Asia and Europe). This dataset is challenging as it includes interactions between vehicles, different environments and multiple potetially plausible predictions.

Following the same rule as previous works [22,24], the dataset is split into 70% training, 10% validation, and 20% testing. In each 8-s trajectory data clip, the last 3-s data are used to predict the future 5-s trajectories. We define the interactive range amongst vehicles as 180 feet (±90 feet).

All experiments are carried out on a platform with an Intel i7-10700K CPU and an Nvidia GeForce RTX 3080 GPU. To evaluate the models, root mean squared error (RMSE) in meters is adopted as the performance metric:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}\sum_{t=t_{obs}+1}^{t_{pred}}((x_i^t - \hat{x}_i^t)^2 + (y_i^t - \hat{y}_i^t)^2)}{N \cdot (t_{pred} - t_{obs})}}. \tag{18}$$

where N denotes the number of vehicles within the interactive range.

4.1. Quantitative Results and Analysis

4.1.1. Baselines

Our proposed model RRN is compared with several baseline models as follows:

- **S-LSTM [23]**: An influential method based on a social pooling model.
- **CS-LSTM [24]**: A model using a convolutional social pooling structure to learn vehicle interactions.
- **MHA-LSTM [5]**: A model based on a multi-head attention mechanism for trajectory prediction.
- **MATF GAN [47]**: A model using convolutional neural network and generative adversarial network to learn social interaction.
- **GRIP [48]**: A model using graph networks to simulate interactions between multiple agents.
- **MFP [22]**: A state-of-the-art model which learns semantically latent variables for trajectory prediction.

4.1.2. Quantitative Evaluation

Tables 1 and 2 show the RMSE results (in meters) of comparison between various models on the NGSIM and INTERACTION datasets. RRN achieves the minimum values on each evaluation point and obtains a relevant improvement of accuracy at the end point, compared to the state-of-the-art model MFP.

Table 1. RMSE (in meters) results of experiments on the NGSIM. We compare the performance of several baseline models with the RRN over the 5-s prediction horizon.

Time	S-LSTM	CS-LSTM	MHA-LSTM	MATF GAN	GRIP	MFP	RRN
1 s	0.65	0.64	0.56	0.66	0.64	0.52	**0.49**
2 s	1.31	1.27	1.22	1.34	1.13	1.11	**1.09**
3 s	2.16	2.09	2.01	2.08	1.80	1.79	**1.76**
4 s	3.25	3.10	3.00	2.97	2.62	2.59	**2.54**
5 s	4.55	4.37	4.25	4.13	3.60	3.53	**3.44**

Table 2. RMSE (in meters) results of experiments on the INTERACTION. We compare the performance of several baseline models with the RRN over the 5-s prediction horizon.

Time	S-LSTM	CS-LSTM	MHA-LSTM	MATF GAN	GRIP	MFP	RRN
1 s	0.33	0.16	0.16	0.16	0.13	0.12	**0.11**
2 s	0.76	0.72	0.69	0.67	0.56	0.55	**0.52**
3 s	1.77	1.77	1.65	1.60	1.34	1.31	**1.31**
4 s	3.42	3.22	3.96	3.72	2.45	2.42	**2.38**
5 s	5.46	4.96	4.61	4.42	3.86	3.84	**3.69**

We also present the error distributions of trajectory prediction based on the RRN model as illustrated in Figure 3. The box-plots shows the minimum, first (lower) quartile, median, third (upper) quartile, and maximum of Euclidean distances between all predicted values and ground truth on the testing set.

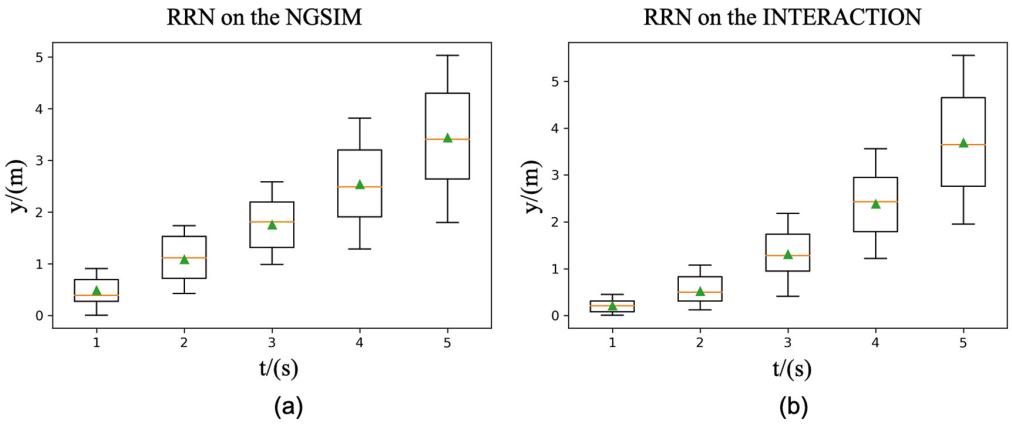

Figure 3. Error distributions of trajectory prediction on the NGSIM and INTERACTION, respectively. The x-axis denotes 5 evaluation points as 1 s, 2 s, 3 s, 4 s, and 5 s; (**a**) Box-plot of RMSE based on RRN on the NGSIM dataset; (**b**) Box-plot of RMSE based on RRN on the INTERACTION dataset.

In order to evaluate the performance toward perturbation occurrences, we add random noise based on a Gaussian distribution $\mathcal{N}(0, \sigma)$ on the history trajectories. σ is set to different values, and the corresponding results are shown in Figure 4. We can see that our method surpasses CS-LSTM, which is the first model using social modeling in the prediction of vehicle trajectories. Wherein, the bias is calculated as the following function. We make a statistic of bias on 5 evaluation points.

$$\text{Bias} = \frac{RMSE_{noise} - RMSE_{raw}}{RMSE_{raw}}. \tag{19}$$

In addition, the computation cost of our method is about 5.1 ms on average, compared to the typical models such as CS-LSTM (about 3 ms) and MFP (about 5 ms). This is still an acceptable and reasonable cost for an autonomous driving system.

4.1.3. Ablation Study

An ablation study is carried out to evaluate the effect of the anchor generator and RLS module in trajectory refinement. Two basic models are used here, i.e., Vanilla LSTM, a simple LSTM framework without anchor generator and RLS module, and Attention LSTM, an LSTM model with attention to learning social interaction. We compare the parametric-learning RLS model with the original one over the 5-s prediction horizon by using RMSE as shown in Table 3.

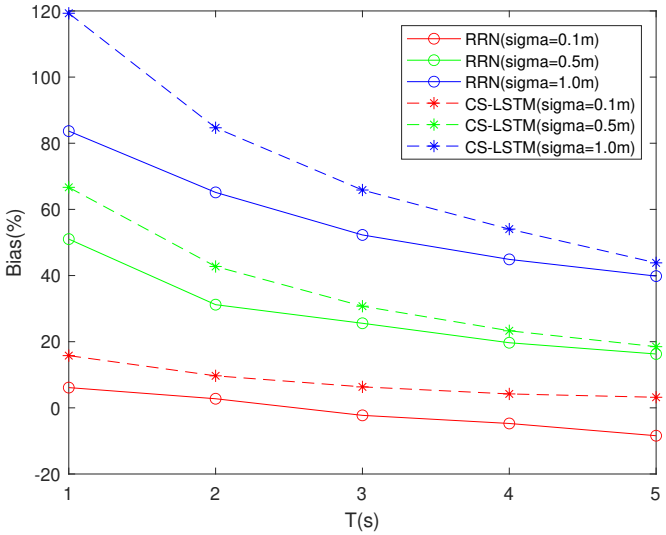

Figure 4. Results of perturbation occurrence. We make a comparison of our model RRN with CS-LSTM.

Table 3. RMSE (in meters) results of ablation study. Vanilla LSTM (+RLS) and Attention LSTM (+RLS) stand for Vanilla LSTM with parametric-learning RLS and Attention LSTM with parametric-learning RLS.

Time	Vanilla LSTM	Vanilla LSTM (+RLS)	Attention LSTM	Attention LSTM (+RLS)
1s	0.65	**0.64**	0.56	**0.56**
2s	1.58	**1.56**	1.21	**1.20**
3s	**2.77**	2.78	1.96	**1.94**
4s	4.26	4.27	2.86	**2.78**
5s	6.11	**6.01**	4.01	**3.89**

Through addition of the anchor generator and RLS module, the new models reduce the RMSE at most time steps. In particular, there are improvements at the final point by 1.64% on Vanilla LSTM and 3.0% on Attention LSTM. Although the improvement appears to not be very salient in the first 4-s of Vanilla LSTM and Attention LSTM, it has proved the feasibility of aggregating various scale information.

The results indicate that our method can improve the performance of rollout trajectory prediction effectively. The anchor generator and RLS module are useful for reducing error accumulation and capturing global motion patterns.

4.2. Qualitative Results and Analysis

The comparison between MFP and RRN is illustrated in several scenarios as shown in Figures 5 and 6, which correspond to highways in the NGSIM and roundabouts in the INTERACTION. Trajectories estimated by the RRN model reveal that anchors work in most cases, while MFP sometimes generates a trajectory with large error due to the rollout mechanism. RRN achieves remarkable refinement over the predicted trajectories.

In many cases, RRN yaws as well at the first few time steps. Although the RRN corrects the yawing to a certain extent over time. Particularly, as shown in Figure 5d, there exists an obvious wheeling in the trajectories by the RRN model. This should be attributed to the effect of the anchor generator and RLS module, which help to capture and balance between local and global motion patterns. Nonetheless, there is still abundant space for further work in avoiding mode-averaging to render the trajectory plausible.

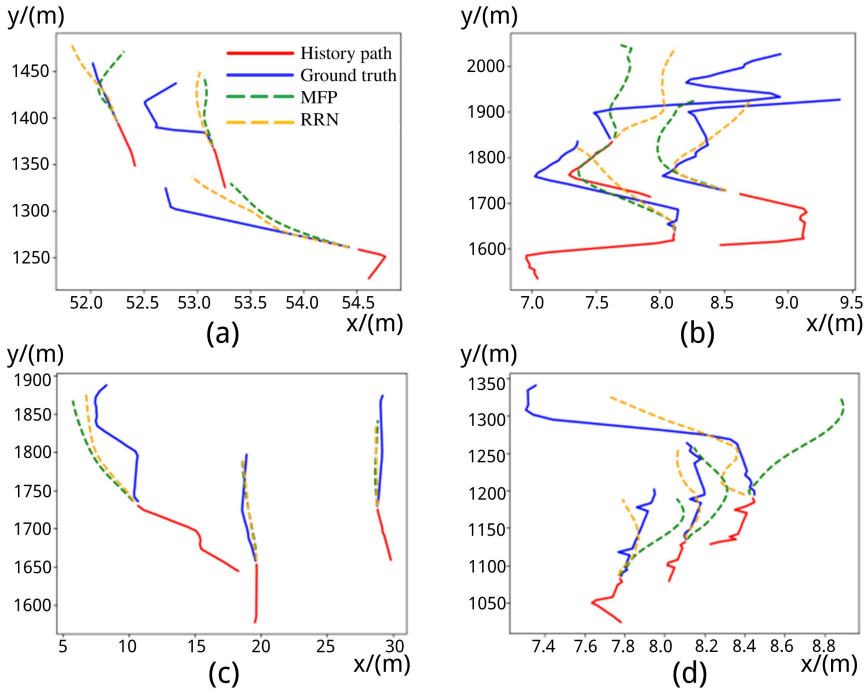

Figure 5. Examples of trajectory visualization on the NGSIM. We present the prediction results (in meters) of MFP and RRN in several cases. The red line denotes the history trajectories. The ground truth is shown by the blue line. The predicted trajectories of MFP and RRN are, respectively, plotted in the green and orange dotted lines. Subfigures (**a**–**d**) illustrate predicted trajectories of 4 various cases on the NGSIM dataset.

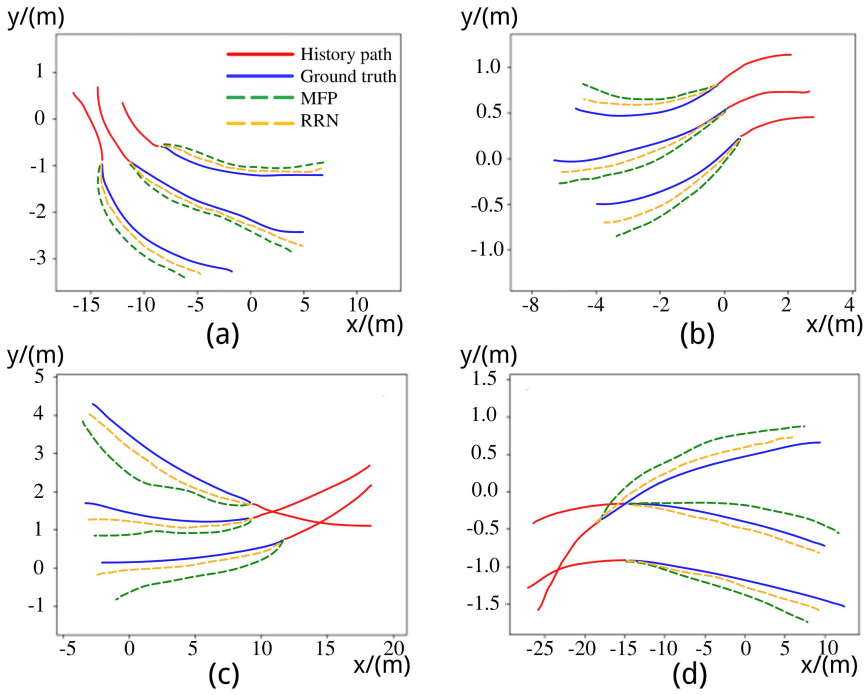

Figure 6. Examples of trajectory visualization on the INTERACTION. We present the prediction results (in meter) of MFP and RRN in the roundabout conditions. Subfigures (**a**–**d**) illustrate predicted trajectories of 4 various cases on the INTERACTION dataset.

5. Conclusions

This study is launched to improve the performance of rollout methods of trajectory prediction on road scenes. We propose a recursive least-squares-based refinement network to capture both local and global motion patterns of surrounding vehicles and refine the rollout trajectories. A data-driven recursive least-squares method is extended in a plausible deep learning manner.

Results of experiments show a promising improvement in trajectory prediction accuracy towards the existing rollout methods on the real-world NGSIM and INTERACTION datasets. Ablation studies have proved the effectiveness of anchor generators and RLS modules on rollout methods. Our work reported here sheds new light on the integration of an interpretable filtering model and the powerful deep learning method.

This research could proceed further by extending RLS to Kalman filter [49]. The PSO with a deep learning model such as CNN and RNN presents another research direction [50]. Furthermore, the next step is to bring the distribution of other objects, i.e., pedestrians and cyclists into the training and testing dataset. This could allow the model of trajectory prediction work in different traffic conditions.

Author Contributions: Conceptualization, S.L., Q.X. and X.L.; methodology, S.L., Q.X. and X.L.; software, S.L. and Q.X.; validation, Q.X., X.L.; formal analysis, S.L., Q.X. and X.L.; investigation, S.L., Q.X. and X.L.; resources, X.L. and W.Z.; data curation, S.L., Q.X., X.L. and D.S.; writing—original draft preparation, Q.X., L.S. and X.L.; writing—review and editing, X.L., L.S. and Q.X.; visualization, S.L.; supervision, X.L. and W.Z.; project administration, X.L., W.Z. and D.S.; funding acquisition, X.L. and W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Key R&D Program of China under Grant 2021YFB1600501, National Natural Science Foundation of China under Grant 61906038, and the Fundamental Research Funds for the Central Universities under Grant 2242021R41184.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dixit, A.; Kumar, C.R.; Allam, Z. Safety and Risk Analysis of Autonomous Vehicles Using Computer Vision and Neural Networks. *Vehicles* **2021**, *3*, 595–617. [CrossRef]
2. Lefèvre, S.; Vasquez, D.; Laugier, C. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech J.* **2014**, *1*, 1. [CrossRef]
3. Paden, B.; Čáp, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–35. [CrossRef]
4. Schmidhube, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
5. Messaoud, K.; Yahiaoui, I.; Verroust-Blondet, A.; Nashashibi, F. Attention Based Vehicle Trajectory Prediction. *IEEE Trans. Intell. Veh.* **2021**, *6*, 175–185. [CrossRef]
6. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2013.
7. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. In Proceedings of the Neural Information Processing Systems, Montréal, QC, Canada, 8–11 December 2014.
8. Gao, J.; Sun, C.; Zhao, H.; Shen, Y.; Anguelov, D.; Li, C.; Schmid, C. VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 11522–11530.
9. Cho, K.; Ha, T.; Lee, G.; Oh, S. Deep Predictive Autonomous Driving Using Multi-Agent Joint Trajectory Prediction and Traffic Rules. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, the Venetian Macao, Macau, China, 3–8 November 2019; pp. 2076–2081.
10. Graves, A. Generating Sequences With Recurrent Neural Networks. *arXiv* **2013**, arXiv:1308.0850.
11. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
12. Jo, E.; Sunwoo, M.; Lee, M. Vehicle Trajectory Prediction Using Hierarchical Graph Neural Network for Considering Interaction among Multimodal Maneuvers. *Sensors* **2021**, *21*, 53–54. [CrossRef] [PubMed]

13. Ju, C.; Wang, Z.; Long, C.; Zhang, X.; Chang, D.E. Interaction-aware Kalman Neural Networks for Trajectory Prediction. In Proceedings of the IEEE Intelligent Vehicles Symposium, Las Vegas, NV, USA, 19 October–13 November 2020; pp. 1793–1800.
14. Mercat, J.; Gilles, T.; El Zoghby, N.; Sandou, G.; Beauvois, D.; Gil, G.P. Mul-ti-Head Attention for Multi-Modal Joint Vehicle Motion Forecasting. In Proceedings of the IEEE International Conference on Robotics and Automation, Paris, France, 31 May–1 August 2020; pp. 9638–9644.
15. Vahidi, A.; Stefanopoulou, A.; Peng, H. Recursive least squares with forgetting for online estimation of vehicle mass and road grade: Theory and experiments. *Veh. Syst. Dyn.* **2005**, *43*, 31–35. [CrossRef]
16. Punzo, V.; Borzacchiello, M.T.; Ciuffo, B. On the assessment of vehicle trajectory data accuracy and application to the Next Generation SIMulation (NGSIM) program data. *Transp. Res. Part Emerg. Technol.* **2011**, *19*, 1243–1262. [CrossRef]
17. Zhang, W.; Sun, L.;Wang D.; Shi, H.; Clausse, A.; Naumann, M.; Kümmerle, J.; Königshof, H.; Stiller, C.; de La Fortelle, A.; et al. INTERACTION Dataset: An International, Adversarial and Cooperative Motion Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv* **2019**, arXiv:1910.03088.
18. Cui, H.; Radosavljevic, V.; Chou, F.; Lin, T.; Nguyen, T.; Huang, T.; Schneider, J.G.; Djuric, N. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In Proceedings of the International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019; pp. 2090–2096.
19. Sun, L.; Zhan, W.; Tomizuka, M. Probabilistic Prediction of Interactive Driving Behavior via Hierarchical Inverse Reinforcement Learning. In Proceedings of the International Conference on Intelligent Transportation Systems, Maui, HI, USA, 4–7 November 2018; pp. 2111–2117.
20. Rudenko, A.; Palmieri, L.; Herman, L.; Kitani, K.M.; Gavrila, D.M.; Arras, K.O. Human motion trajectory prediction: A survey. *Int. J. Robot. Res.* **2020**, *39*, 895–935. [CrossRef]
21. Lee, N.; Choi, W.; Vernaza, P.; Choy, C.B.; Torr, P.H.; Chandraker, M. Desire: Distant future prediction in dynamic scenes with interacting agents. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 336–345.
22. Tang, Y.C.; Salakhutdinov, R. Multiple futures prediction. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 15424–15434.
23. Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Li, F.F.; Savarese, S. Social LSTM: Human Trajectory Prediction in Crowd-ed Spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 961–971.
24. Deo, N.; Trivedi, M.M. Convolutional Social Pooling for Vehicle Trajectory Prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–23 June 2018; pp. 1549–1557.
25. Gupta, A.; Johnson, J.; Li, F.F.; Savarese, S.; Alahi, A. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2255–2264.
26. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
27. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In Proceedings of the Advances in Neural Information Processing Systems Workshop on Deep Learning, Montréal, QC, Canada, 8–13 December 2014.
28. Ondruska, P.; Posner, I. Deep tracking: Seeing beyond seeing using recurrent neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 3361–3367.
29. Khosroshahi, A.; Ohn-Bar, E.; Trivedi, M.M. Surround vehicles trajectory analysis with recurrent neural networks. In Proceedings of the IEEE International Conference on Intelligent Transportation Systems, Rio de Janeiro, Brazil, 1–4 November 2016; pp. 2267–2272.
30. Kim, B.; Kang, C.M.; Lee, S.H.; Chae, H.; Kim, J.; Chung, C.C.; Choi, J.W. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In Proceedings of the IEEE International Conference on Intelligent Transportation Systems, Yokohama, Japan, 16–19 October 2017; pp. 399–404.
31. Park, S.H.; Kim, B.; Kang, C.M.; Chung, C.C.; Choi, J.W. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In Proceedings of the IEEE Intelligent Vehicles Symposium, Changshu, China, 26–30 June 2018; pp. 1672–1678.
32. Du, S.; Li, T.; Yang, Y.; Horng, S.J. Multivariate time series forecasting via attention-based encoder–decoder framework. *Neural Comput.* **2020**, *388*, 269–279. [CrossRef]
33. Fang, L.; Jiang, Q.; Shi, J.; Zhou, B. Tpnet: Trajectory proposal network for motion prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 13–19 June 2020; pp. 6797–6806.
34. Mangalam, K.; Girase, H.; Agarwal, S.; Lee, K.H.; Adeli, E.; Malik, J.; Gaidon, A. It Is Not the Journey However, the Destination: Endpoint Conditioned Trajectory Prediction. In Proceedings of the IEEE European Conference on Computer Vision, online-only, 23–28 August 2020; pp. 759–776.
35. Dendorfer, P.; Osep, A.; Leal-Taixé, L. Goal-gan: Multimodal trajectory prediction based on goal position estimation. In Proceedings of the Asian Conference on Computer Vision, Kyoto, Japan, 30 November–4 January 2020; pp. 405–420.

36. Zhao, H.; Wildes, R.P. Where are you heading? Dynamic Trajectory Prediction with Expert Goal Examples. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 7629–7638.
37. Zhao, H.; Gao, J.; Lan, T.; Sun, C.; Sapp, B.; Varadarajan, B.; Shen, Y.; Shen, Y.; Chai, Y.; Schmid, C.; et al. TNT: Target-driven Trajectory Prediction. In Proceedings of the Conference on Robot Learning, London, UK, 8–11 November 2021; pp. 895–904.
38. Gu, J.; Sun, C.; Zhao, H. Densetnt: End-to-end trajectory prediction from dense goal sets. In Proceedings of the IEEE International Conference on Computer Vision, Montréal, QC, Canada, 10–17 October 2021; pp. 15303–15312.
39. Casas, S.; Luo, W.; Urtasun, R. Intentnet: Learning to predict intention from raw sensor data. In Proceedings of the Conference on Robot Learning, Zurich, Switzerland, 29–31 October 2018; pp. 947–956.
40. Feng, X.; Cen, Z.; Hu, J.; Zhang, Y. Vehicle trajectory prediction using intention-based conditional variational autoencoder. In Proceedings of the IEEE Intelligent Transportation Systems Conference, Auckland, New Zealand, 27–29 October 2019; pp. 3514–3519.
41. Li, L.; Yao, J.; Wenliang, L.; He, T.; Xiao, T.; Yan, J.; Wipf, D.; Zhang, Z. GRIN: Generative Relation and Intention Network for Multi-agent Trajectory Prediction. In Proceedings of the Advances in Neural Information Processing Systems, online-only, 7–10 December 2021.
42. Chai, Y.; Sapp, B.; Bansal, M.; Anguelov, D. MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction. In Proceedings of the Conference on Robot Learning, Osaka, Japan, 30 October–1 November 2019.
43. Phan-Minh, T.; Grigore, E.C.; Boulton, F.A.; Beijbom, O.; Wolff, E.M. Covernet: Multimodal behavior prediction using trajectory sets. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 14074–14083.
44. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
45. Ma, X.; Karkus, P.; Hsu, D.; Lee, W.S. Particle Filter Recurrent Neural Networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 5101–5108.
46. Engel, Y.; Mannor, S.; Meir, R. The kernel recursive least-squares algorithm. *IEEE Trans. Signal Process.* **2004**, *52*, 2275–2285. [CrossRef]
47. Zhao, T.; Xu, Y.; Monfort, M.; Choi, W.; Baker, C.; Zhao, Y.; Wang, Y.; Wu, Y.N. Multi-agent tensor fusion for contextual trajectory prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 12118–12126.
48. Li, X.; Ying, X.; Chuah, M.C. GRIP: Graph-based Interaction-aware Trajectory Prediction. In Proceedings of the IEEE Intelligent Transportation Systems Conference, Auckland, New Zealand, 27–30 October 2019; pp. 3960–3966.
49. Kalman, R.E. A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.* **1960**, *82*, 35. [CrossRef]
50. Lei, X.; Dali, D.; Hongpeng, Z.; Jianpu, W.; Zhuoran, Z. UCAV maneuvering trajectory prediction based on PSO-CNN. In Proceedings of the 2021 International Conference on Computer Engineering and Application, Guangzhou, China, 25–27 June 2021; pp. 54–58.