Socially Aware Kalman Neural Networks for Trajectory Prediction

Ce Ju [*] [‡]
juce@baidu.com

Zheng Wang[†] [‡]
wang_zheng@ntu.edu.sg

Xiaoyu Zhang[*]
zhangxiaoyu09@baidu.com

Abstract

Trajectory prediction is a critical technique in the navigation of robots and autonomous vehicles. However, the complex traffic and dynamic uncertainties yield challenges in the effectiveness and robustness in modeling. We propose a data-driven approach *socially aware Kalman neural networks* (SAKNN) where the interaction layer and the Kalman layer are embedded in the architecture, resulting in a class of architectures with huge potential to directly learn from high variance sensor input and robustly generate low variance outcomes. The evaluation of our approach on NGSIM dataset demonstrates that SAKNN performs *state-of-the-art* on prediction effectiveness in a relatively long-term horizon and significantly improves the signal-to-noise ratio of the predicted signal.

Introduction

Forecasting the motion of surrounding vehicles is very critical for the navigation of robots and autonomous vehicles. Imagine that if a surrounding vehicle has an intention to cut in your lane, it is necessary to estimate its trajectory in the following few seconds as a priori for planning a non-intersection path as precisely as possible. In order to improve the safety and efficiency of the autonomous vehicles, a well-designed autonomous driving system should have the ability to predict the future traffic scenes over a relatively long-term horizon.

Generally, forecasting the future trajectory of a vehicle is very challenging. Because it is typically an issue of the complex system which is affected by the variable weather, the complex traffic scenes, the different driving styles of human drivers and even the massive interaction between neighbor vehicles. Recall the previous example that supposes your feedback strategy is to speed up your vehicle in order to prevent your neighbor to cut in your lane, it will dramatically change his original intention in a sudden, and he will then recast his driving path soon. This is a simple traffic interaction scene but mathematically modeling of it is not an easy task.

[*]Baidu Inc. No. 10, Shangdi 10th Street, Beijing, 100085, China.

[†]School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Ave, 639798, Singapore.

[‡]Contributed Equally

In spite of the challenges above, we exploit the following two considerations in our model:

- Socially Aware Interactive Effects: The socially aware interactive effect is cased by the social force which is a measure of the internal motivation of an individual. It is proposed by D.Helbing et al. (Helbing and Molnar 1995) to study the motion trajectories of a group of pedestrians. In our model, we assume the dynamic motion of vehicles in complex traffic is affected by the socially aware interaction between neighbor vehicles.
- Dynamic Uncertainties: Forecasting trajectories of robots or autonomous vehicles is intensely entangled with linear or nonlinear dynamic models since of uncertainties. In our model, the neural network is formulated in a dynamic model-based approach that a filter layer is embedded into.

Inspired by the recent literature of artificial intelligence with the application in the complex system and nonlinear dynamic model (Langford, Salakhutdinov, and Zhang 2009; Krishnan, Shalit, and Sontag 2015), we propose a neural network architecture called Socially Aware Kalman Neural Networks (SAKNN). SAKNN is a deep neural network with an embedding filter layer which has the following three advantages:

- Robustness: The robustness is attributed to the powerful ability of extracting complex abstractions of deep neural network (Deng, Yu, and others 2014), especially the multiple convolution layers in front of SAKNN, to resolve a hierarchy of concepts of massive surrounding sensor data and extract unique features from the data of traffic and vehicle dynamics, while human experts almost impossibly can.
- Effectiveness: After a relatively long-term horizon, SAKNN overwhelms all the baselines in the mean square error (MSE) sense which is attributed to an embedding filter structure. Actually, we embed a filter layer into SAKNN as an approximation structure of the Kalman filter (KF). In training, the filter layer recursively bends the weights of the neural network to reach the optimal estimation of the underlying states in the MSE sense.
- Smoothness: In the light of the embedding filter structure of SAKNN, we set a smoothing regularization to bend the weights of the neural network in which we directly ex-

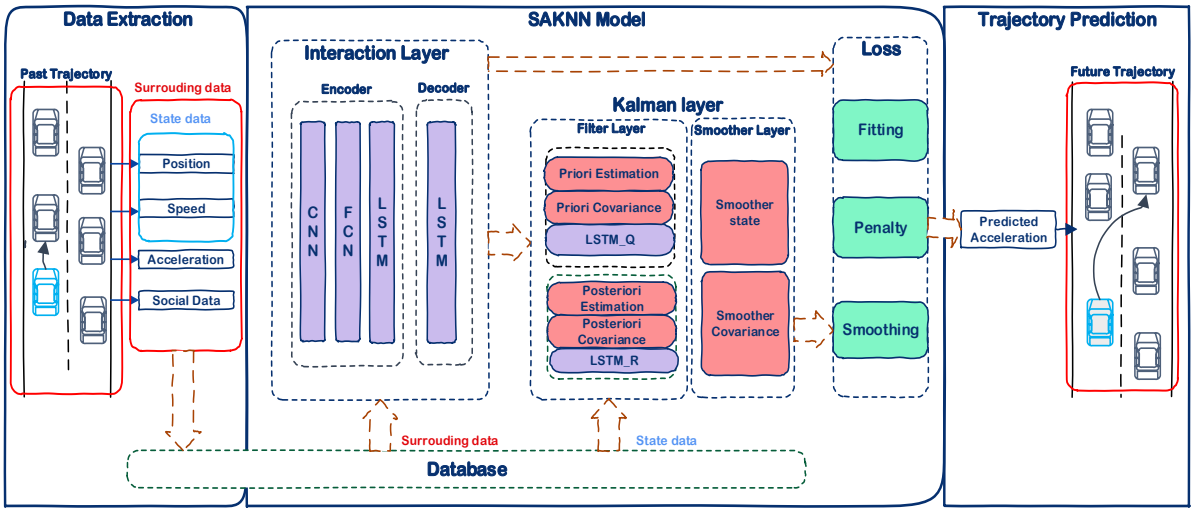

Figure 1: Illustration of **SAKNN Model:** The graph is composed of data extraction part, which we extract four categories of data in history from NGSIM data set, and the SAKNN model part in which we feed data to the interaction layer to get predicted acceleration and then recursively filtering and smoothing raw NGSIM data and predicted acceleration to estimate underlying states in training. In the trajectory prediction part, the predicted acceleration generated by the interaction layer is integrated to predicted trajectories of vehicles with respect to motion formula. See section Model for details.

ecute the Kalman smoother backwardly on the predicted trajectory for smoothing.

In the experimental evaluation, we apply SAKNN to the Next Generation Simulation (NGSIM) data set (Colyar and Halkias 2007) and the empirical results demonstrate that SAKNN predicts more precisely in a relatively long-term horizon and significantly improves the signal-to-noise ratio of the predicted signal.

Related Work

We adopt the categories of methods of the trajectory prediction proposed by S.Lefevre et al. (Lefèvre, Vasquez, and Laugier 2014) in which the abstraction of their classifications is gradually increasing, i.e. physics-based motion model, maneuver-based motion model and interaction-aware motion model. Actually, SAKNN is a combination of physics-based motion model and interaction-aware motion model.

Dynamic Uncertainties

The uncertainties need to be taken into account for modeling of the trajectory prediction more precisely. Typically, the KF (Kalman 1960) is applied to model this uncertainty and recursively estimate the states of robots or autonomous vehicles from noisy sensor measurements. For a more complete review of state estimation, we refer the reader to standard references (Bishop, Welch, and others 2001; Thrun, Burgard, and Fox 2005; Simon 2006). In recent literature, the neural network approach for state estimation has been explored largely (Bobrowski et al. 2008; Wilson and Finkel 2009). H.Coskun et al.(Coskun et al. 2017) train triple long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) to learn the motion model, the process noise, and the measurement noise respectively in order to estimate the human pose in a camera image. T.Haarnoja et al. (Haarnoja et al. 2016) propose a discriminative approach to state estimation in which they train a neural network to learn features from highly complex observations and then filter with the learned features to output underlying states.

Socially Aware Model

Interaction layer takes into account the influences of neighboring pedestrians or vehicles. Comparing with the traditional dynamic model-based approach of modeling these dynamic features, we mainly refer to the recent work in the neural network approach. A.Alahi (Alahi et al. 2016) proposes a deep neural network model to predict the motion dynamics of pedestrians in crowded scenes, in which they build a fully connected layer called social pooling to learn the social tensor and the interaction between each pedestrian. However, this approach is designed for the motion of the pedestrians and not effective in complex traffic configurations. Another work of motion prediction is from (Deo and Trivedi 2018). In their work, they extract social tensor by a convolutional social pooling layer and then feed the social tensor to a maneuver based model of trajectory prediction. A special version of the prediction model is applying imitation learning approach to learn human driver behavior in the multi-agent setting (Bhattacharyya et al. 2018; Kuefler et al. 2017). The learned policies are able to generate the future driving trajectories which are better to match those of human drivers and can also interact with neighbors in a more stable manner over long horizons in the multi-

agent setting.

Background

Notation

In the following sections, we use alphabets t_0, t_1, T to represent the Kalman filter starting time, the Kalman filter terminal time and the Kalman prediction terminal time respectively. And, the subscript s:t represents a history of data or states from Time s to Time t. We also use $\mathcal{W}; b$ to represent the weights of the neural networks Interaction$_N$, LSTM$_R$ and LSTM$_Q$ in our architecture.

Formulation of Our Problem

We consider the basic kinematic motion formula. Let p_t be the position at Time t in the trajectory of the vehicles. The motion formula is as follows,

$$p_t = p_{t-1} + v_t \cdot \Delta t + \frac{1}{2} a_t \cdot \Delta t^2 + \mathcal{O}(p_{t-1}^{(3)}), \tag{1}$$

where we use big $\mathcal{O}$ notation to simply represent the higher order term of Taylor's expansion of position. Our approach aims to predict trajectories of vehicles in two sequential steps. The first step is to predict acceleration generated by our neural network SAKNN whose input is the sensor information from perception space $\mathcal{S}$ which mainly includes a history of positions, velocities, accelerations and social data; In the second step, we integral the predicted acceleration with respect to kinematic motion formula (1).

In the next section, we create a new model for tracking this setting. And, in the rest of this section, we provide a small overview of the Kalman filter and smoother which play an important role in our model. The KF is an optimal state estimator in the MSE sense with a linear dynamic model and Gaussian noise assumptions. Suppose we denote the state as x_t, the control as u_t and observation as z_t, the KF is written as the process model and the measurement model

$$\begin{aligned} x_t &= F \cdot x_{t-1} + B \cdot u_{t-1} + w_t \qquad & w_t &\sim \mathcal{N}(0,\, Q_t) \\ z_t &= H \cdot x_t + v_t & v_t &\sim \mathcal{N}(0,\, R_t), \end{aligned}$$

where the matrices F, B and H are constant and process noise Q_t and measurement noise R_t are covariance matrices of zero mean Gaussian white noise with respect to time.

The KF adopts a recursive way to estimate underlying states of the above equation in two steps. Typically, the prediction and the update phases alternate until the observations are unavailable. In the prediction step, the current states x_t^- and the error covariance matrix P_t^- is estimated by the process model which is independent of current measurement

$$\begin{aligned} x_t^- &= F \cdot x_{t-1} + B \cdot u_{t-1} \\ P_t^- &= F \cdot P_{t-1} \cdot F^T + Q_t \end{aligned}$$

In the update step, once received the current observations, Kalman gain K_t, the prior estimation x_t^- and the error covariance matrix P_t^- is calculated by the measurement model,

$$\begin{aligned} K_t &= P_t^- \cdot H^T \cdot (H \cdot P_t^- \cdot H^T + R_t)^{-1} \\ x_t &= x_t^- + K_t \cdot (z_t - H \cdot x_t^-) \\ P_t &= (I - K_t \cdot H) \cdot P_t^- \end{aligned}$$

After the KF, we backwardly carry out the recursive steps of Kalman smoother as follows

$$\begin{aligned} L_t &= P_t \cdot F^T \cdot {P_{t+1}^-}^{-1} \\ \tilde{x}_t &= x_t + L_t \cdot (\tilde{x}_{t+1} - x_{t+1}^-) \\ \tilde{P}_t &= P_t^- + L_t \cdot (\tilde{P}_{t+1} - P_t^-), \end{aligned}$$

where the new symbol $\tilde{x}_t$ and $\tilde{P}_t$ are the smoothed states and the smoothed error covariance matrix.

Model

In this section, we present our model of Socially Aware Kalman Neural Network (SAKNN). See Figure 1 for the graph of the detailed architecture.

Architecture of Interaction Layer

SAKNN takes into account a slightly influence of social force between vehicles and simulate such dynamical features. In contrast to the Helbing-Molnar-Farkas-Vicsek social force model (Helbing, Farkas, and Vicsek 2000), social force between vehicles is the intention of drivers not to collide with the others or static obstacles.

According to Newton's 2nd Law in physics, acceleration is directly proportional to force and thus we decide to pick acceleration to be the output of SAKNN. Thus, the interaction operator is from the perception space $\mathcal{S}$ to the control space $\mathcal{A}$, i.e.

$$\text{Interaction}_N(\cdot) : \mathcal{S} \to \mathcal{A}$$

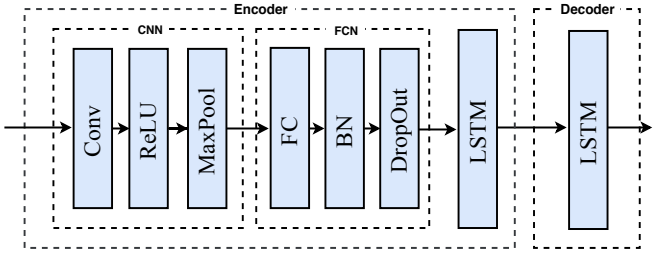

Figure 2: Illustration of the **Interaction Layer**: Our main architecture of interaction layer is composed of the encoder part and the decoder part. In the encoder part, we sequentially build CNN layers which can be regarded as social tensor extractor and FCN layers which can be regarded a mixer of social tensor, and then we merge the deep features into the encoder LSTM and the decoder LSTM.

The architecture of the interaction layer refers to Figure 2. Specifically, the input of interaction layer is a history of the recorded velocities v, positions p, accelerations a, widths w and lengths l of vehicles, and distances d between two vehicles and the interaction layer is written as

$$A_{t+1:t+T-t_1} = \text{Interaction}_N(v_{t_0:t}, p_{t_0:t}, a_{t_0:t}, w_{t_0:t}, l_{t_0:t}, d_{t_0:t})$$

Inspired by the work of Helbing et al. (Helbing and Molnar 1995), we also feed the hand-crafted repulsive interaction force

$$e_{t_0:t} := \exp\big((v_i + v_j) \cdot \Delta t - d\big)_{t_0:t}$$

to our neural network, where the subscript i and j represent two adjacent vehicle i and vehicle j.

Multi-Agent Kalman filter and smoother

We denote interaction layer as Interaction_N, process noise neural network as LSTM_Q and measurement noise neural network as LSTM_R.

The diagram of multi-agent KF of N-vehicle motion refers to Figure 3 and we write it in a recursive way as follows,

$$x_t = F \cdot x_{t-1} + B \cdot \text{Interaction}_N(s_t) + w_t$$
$$z_t = x_t + v_t,$$

where s_t represents surrounding data; The state vector x_t and measurement vector z_t consist of positions and velocities respectively written as

$$x_t := \begin{bmatrix} p_t^1 \\ v_t^1 \\ \dots \\ p_t^N \\ v_t^N \end{bmatrix}_{2N\times 1} \quad z_t := \begin{bmatrix} \bar{p}_t^1 \\ \bar{v}_t^1 \\ \dots \\ \bar{p}_t^N \\ \bar{v}_t^N \end{bmatrix}_{2N\times 1}.$$

The motion matrix F and control matrix B are determined by motion formula and written as

$$F := \begin{bmatrix} 1 & \Delta_t & & & & & \\ & 1 & & & & & \\ & 1 & \Delta_t & & & & \\ & & 1 & & & & \\ & & & \dots & & & \\ & & & \dots & & & \\ & & & & \Delta_t & & \\ & & & & 1 & & \\ & & & & 1 & \Delta_t & \\ & & & & & 1 & \end{bmatrix}_{2N\times 2N}$$

$$B := \begin{bmatrix} \frac{1}{2}\Delta_t^2 & & \\ \Delta_t & & \\ & \dots & \\ & \dots & \\ & & \frac{1}{2}\Delta_t^2 \\ & & \Delta_t \end{bmatrix}_{2N\times N}$$

And, process noise matrix w_t obeys Gaussian $w_t \sim \mathcal{N}\big(0, \text{LSTM}_Q(x_{t_0:t}^-)\big)$ and measurement noise matrix v_t also obeys Gaussian $v_t \sim \mathcal{N}\big(0, \text{LSTM}_R(z_{t_0:t})\big)$.

The prediction step in this N-agent Kalman filter is then written as

$$x_t^- = F \cdot x_{t-1} + B \cdot A_{t-1}$$
$$P_t^- = F \cdot P_{t-1} \cdot F^T + Q_t$$

And, the update step of Kalman filter is written as

$$K_t = P_t^- \cdot (P_t^- + R_t)^{-1}$$
$$x_t = x_t^- + K_t \cdot (z_t - x_t^-)$$
$$P_t = (I - K_t) \cdot P_t^-,$$

where vector A_{t-1}:=$\text{Interaction}_N(s_t)$ is a vector of the predicted acceleration of vehicles. And, R_t and Q_t are the diagonal matrix of the output of $\text{LSTM}_R(\cdot)$ and $\text{LSTM}_Q(\cdot)$ module respectively.

Finally, the recursive step of Kalman smoother is written as

$$L_t = P_t \cdot F^T \cdot (P_{t+1}^-)^{-1}$$
$$\tilde{x}_t = x_t + L_t \cdot (\tilde{x}_{t+1} - x_{t+1}^-)$$
$$\tilde{P}_t = P_t^- + L_t \cdot (\tilde{P}_{t+1} - P_t^-),$$

where L_t is smoothing matrix and $\tilde{x}_t$ and $\tilde{P}_t$ are the smoothed state and the smoothed error covariance matrix.

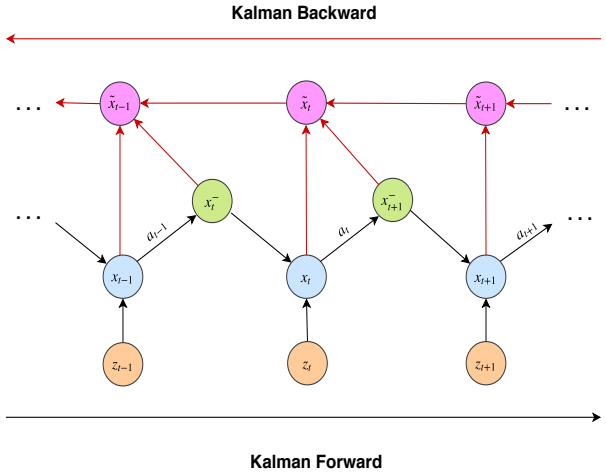

Figure 3: Illustration of the **Kalman Layer**: We execute Kalman filter and smoother in the Kalman layer in which we predict priori and update posteriori forwardly, and smooth them backwardly. Note that the acceleration from prior to posterior in this figure is the predicted acceleration from the interaction layer. In the forward process, which is represented by the black arrow, the current underlying x_t is updated by current observation z_t and previous priori x_t^-. In the backward process, which is represented by the red arrow, the smoothed state $\tilde{x}_t^-$ is obtained by the current state x_t, the next priori x_{t+1}^- and the smoothed state $\tilde{x}_{t+1}^-$.

Architecture for Noise Model

To model the sequential data, LSTM (Hochreiter and Schmidhuber 1997) and GRU (Cho et al. 2014) are the most popular deep neural network architectures till now. These two gated-structure models overwhelm all models on the metric of recognition accuracy in quite a few fields e.g. natural language text compression, unsegmented connected handwriting recognition, and speech recognition (Graves et al. 2009; Graves, Mohamed, and Hinton 2013).

In SAKNN, we train two gated-structure neural networks to learn the time-varying sensor noise in filter layer, i.e.

Process model noise covariance: $\text{LSTM}_Q(x_{t_0:t}^-)$;

Measurement model noise covariance: $\text{LSTM}_R(z_{t_0:t})$.

Loss Function and Regularization

The loss in SAKNN is set for three goals, to smooth the recorded predicted states in the KF, to fit the predicted acceleration of network and constrain of future predicted states in the KF.

- Fitting of accelerations: At each time step of the KF, the interaction layer will produce current and future acceleration. We fit the predicted acceleration and its corresponding ground truth a^{GT} in square L2 norm. In practice, the ground truth of acceleration is directly from raw sensor data a^{GT}.

  $$\mathcal{L}_{\mathcal{F}}(\mathcal{W};b) := \mathbb{E}\ ||\big(a_{t+1:t+T-t_1}(\mathcal{W};b)\big) - \big(a^{GT}_{t+1:t+T-t_1}\big)||^2_2,$$

  where we take an expectation of the fitting loss to normalize the effects of a big number of Kalman time-step results.

- The Penalty of prediction: After an integration of acceleration with respect to the kinematic motion formula (1), we give a penalty in square L2 norm to the predicted trajectory in order to constrain the final displacement. $\mathcal{P}(z,a)$ representing the kinematic motion formula starts from observation state z and iterates with the predicted acceleration a. In practice, the ground truth of positions is directly from raw sensor data z^{GT}.

  $$\mathcal{L}_{\mathcal{P}}(\mathcal{W};b) := \mathbb{E}\ ||\mathcal{P}\big(z_t, a_{t+1:t+T-t_1}(\mathcal{W};b)\big) - z^{GT}_{t+1:t+T-t_1}||^2_2,$$

  where we take an expectation of the penalty loss to normalize the effects of a big number of Kalman time-step results.

- Smoothing of predicted states: At each time step of the KF, the filter layer outputs a predicted state $x^-(\mathcal{W};b)$. We collect the predicted states and smooth them by the Kalman smoothing backward step. We enlarge the smoothed trajectory and raw sensor data in order to bend the weights of the neural network. The smoothing regularization is defined as,

  $$\mathcal{R}_{\mathcal{S}}(\mathcal{W};b) := \exp\big(-||\mathcal{S}\big(x^-_{t_0+1:t_1+1}(\mathcal{W};b)\big) - z_{t_0+1:t_1+1}||^2_2\big),$$

  where $\mathcal{S}(\cdot)$ is the Kalman smoothing operator.

The total loss function is composed of the above two losses and regularization,

$$\mathcal{L}(\mathcal{W};b) := \alpha_1 \cdot \mathcal{L}_{\mathcal{F}}(\mathcal{W};b) + \alpha_2 \cdot \mathcal{L}_{\mathcal{P}}(\mathcal{W};b) + \alpha_3 \cdot \mathcal{R}_{\mathcal{S}}(\mathcal{W};b),$$

where α_1,α_2,α_3 are hyperparameters.

Experiment

Dataset

We evaluate our approach on public datasets US Highway 101 (US-101) (Colyar and Halkias 2007) and Interstate 80 (I-80) (Lu and Skabardonis 2007) of NGSIM program. NGSIM program collects detailed vehicle trajectory data on southbound US101 and eastbound I-80 with a software application called NG-VIDEO to transcribe the vehicle trajectory data from the video. The statistics of raw data of US-101 and I-80 are shown in Table 1.

Dataset	Study area	lanes	Time span	Sampling rate
US-101	2,100 feet	6	3×15 min	10 Hz
I-80	1,640 feet	6	3×15 min	10 Hz

Table 1: Dataset Statistics

The training set and testing set are composed of 100,000 frames of raw data extracted from the NGSIM data set. 70% of them are for training and the rest for testing. In particular, we use the raw data for training but filter it for the evaluation. We extract raw data from NGSIM in the following way. We align raw data by the timestamp and group vehicles by one host and five surrounding vehicles on the adjacent traffic lanes. We then set 12 seconds as a window size of experiments in which the first 5 second's trajectory is for a track history and the rest is for the prediction horizon.

Evaluation Metrics

In this subsection, we propose multiple metrics to verify the effectiveness, the robustness and the smoothness of SAKNN. In particular, we filter the raw data from NGSIM to be the ground truth.

- Effectiveness: The two regular metrics in trajectory prediction are average displacement error (ADE) and final displacement error (FDE). ADE is the root mean squared prediction error accumulated by the displacement error between predicted position and the ground truth in each time step. And, FDE takes into account the final displacement error. A small ADE or FDE means that the model predicts well.
- Robustness: We test SAKNN on the general data set and the lane changing data set. In each data set, we compute ADE and FDE to evaluate the robustness. A robust model is corresponding to a small number in both ADE and FDE.
- Smoothness: We use signal-to-noise ratio (SNR) as a metric to quantitatively compute how the smoothness of our predicted signal is. SNR is a measure to compare the level of the underlying signal and the level of background noise.

Baselines

We compare the following baselines with SAKNN.

- *Constant Velocity (CV)*: The basic kinematic motion formula with a constant velocity.
- *Vanilla-LSTM (V-LSTM)*: Referring to (Park et al. 2018), V-LSTM feeds in track history and generates the future trajectory sequence.
- *Acceleration-LSTM (ACC-LSTM)*: We propose it to compare with SAKNN in which LSTM takes the track history and predicts the future acceleration. The predicted trajectory is accumulated by the predicted acceleration according to the basic kinematic motion formula.
- *Convolutional Social Pooling-LSTM (CS-LSTM)*: CS-LSTM is a maneuver based model generating a multi-modal predictive distribution (Deo and Trivedi 2018).

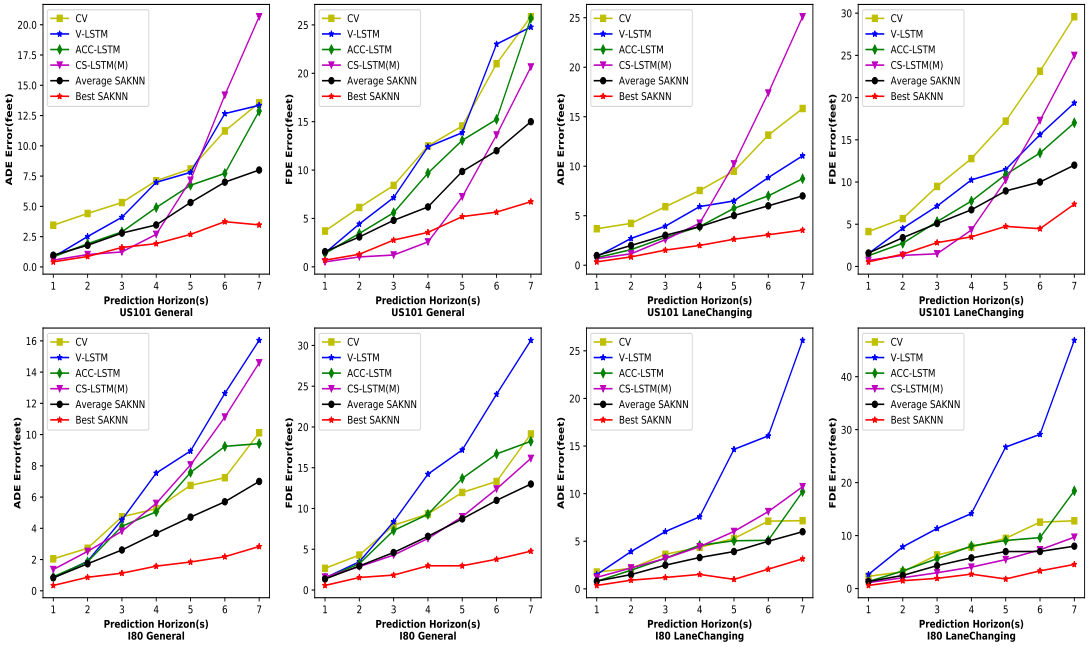

Figure 4: Illustration of the Performance of SAKNN: The data sets labeled by *General* include all the traffic scenes. And, the data sets labeled by *LaneChanging* only include the changing lane traffic scenes.

Results

Figure 4 shows the effectiveness and robustness of SAKNN in both general and lane changing data set. We train SAKNN in almost 500 independent trails and over 300 training episodes in each case until it converges. And, we take an average of ADEs and FDEs of all vehicles to compare with the results of multiple baselines, since SAKNN predicts the future trajectories of a group of six vehicles together. The average of the performance of six vehicles in SAKNN is called Average SAKNN.

Hyperparameters of baselines are fine-tuned to obtain a good score on the validation sets. Another two state-of-the-art algorithms of trajectory prediction *Social Attention LSTM* (Vemula, Muelling, and Oh 2017) and *Social LSTM* (Alahi et al. 2016) is taken into account, but they are designed for the motion of the pedestrians and thus not effective in the complex traffic configurations. Actually, our results differ one order of magnitude, and we decide not to put their results with SAKNN in our figure.

In Figure 4, the Average SAKNN is a little better or slightly worse than the ACC-LSTM and CS-LSTM algorithms in a short-term prediction (1 to 4 seconds). However, the Average SAKNN is superior to all others after 4 seconds and the advantages increase heavily as time propagates. We also observe that the Average SAKNN performs well both in general and lane changing data set. The observations verify that the Average SAKNN indeed has merits in the effectiveness and the robustness, better than baselines. In particular, the Average SAKNN reduces 30% to 60% displacement errors of the constant velocity model in any situations.

In particular, we pick the best performance of six vehicles in SAKNN to be the Best SAKNN. The vehicle which is called Best SAKNN varies in different data sets but overwhelms the baselines substantially.

Qualitative Analysis and Discussion

In this section, we will qualitatively analyze the deep relationship between performance and structure of SAKNN.

- *Sensitivity of Hyperparameters in Architecture*: SAKNN has tens of hyperparameters from the interaction layer to the Kalman layer. The experimental results show that the coefficients of the loss terms and regularization will influence our results. In practice, we take into account the orders of magnitude of the loss terms and regularization and prefer to give fitting term importance.
- *Sensitivity of Hyperparameters in Filtering*: In the evaluation step, we smooth the trajectories from NGSIM by the Savitzky-Golay filter with some window size as a hyperparameter. In fact, ADE and FDE are truly sensitive to this hyper-parameter in the experiments. However, suppose we fix any window size for the filtering of the trajectories, SAKNN is always superior to other baselines in performance.

- *Effects of the Interaction Layer and the Kalman Layer in SAKNN*: The interaction Layer provides a relatively underlying acceleration with low variance Gaussian white noise. And, the Kalman layer is indeed effective for smoothing the predicted acceleration in a slight degree and further improve the signal-to-noise ratio in experiments. See Table 2. Actually, the effect of the Kalman layer in SAKNN is not inferior to it of the Kalman filter and smoother in robotics.

Weights	Acceleration x (dB)	Acceleration y (dB)
Ground Truth	-1.0753	-1.4390
0.0/0.9/0.1	5.9033	5.4834
0.1/0.8/0.1	2.8446	7.4998
0.2/0.7/0.1	11.0210	15.8976
0.3/0.6/0.1	7.9355	16.7044
0.4/0.5/0.1	12.9703	10.0643

Table 2: Signal-to-Noise Ratio of SAKNN: The weight column includes the smoothing/fitting/penalty weights. We randomly pick experiment samples from US-101 data set and compute the average SNR of the samples for the future 20 time steps. We raise the smoothing weight in each experiment and find that the average SNR is increasing quickly but irregularly. The results in the table demonstrate the Kalman layer and smoothing term has effects in our model.

- *Predictive Ability of Regression Model*: A big doubt in our approach is if it is possible to learn acceleration with any architecture of the neural network. The experimental results demonstrate a lack of patterns of the predicted acceleration after enough long time for training. We then create a heuristic method to turn the regression problem to be a classification problem and much tractable. Specifically, we divide the range of acceleration and put them into range boxes which cover the whole range of acceleration of all vehicles. Then, we set $\mathcal{L}_{\mathcal{F}}(\mathcal{W}; b)$ to be the loss of predicted range box number with the box number of the ground truth.

Conclusions and Future Work

In this work, we propose a new model SAKNN of trajectory prediction in which we take into account the two intractable challenges socially aware interactive effects and dynamic uncertainties in robotics and autonomous driving. We embed the interaction layer and the Kalman layer into the architecture of SAKNN to exploit the two challenges respectively. In an extensive set of experiments, SAKNN outperforms in the effectiveness, the robustness and the smoothness than baseline models and achieve state-of-the-art performance on US-101 and I-80 data sets. Further work will extend SAKNN to a probabilistic formulation and combine SAKNN with a maneuver-based model in which road topology and more of the traffic information will be taken into account as a priori.

Acknowledgement

We would like to thank Baidu Apollo community. Especially, we thank the L3 prediction lead Yizhi Xu for his support and some helpful suggestions and discussions in the techniques of the motion prediction.

References

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

[Bhattacharyya et al. 2018] Bhattacharyya, R. P.; Phillips, D. J.; Wulfe, B.; Morton, J.; Kuefler, A.; and Kochenderfer, M. J. 2018. Multi-agent imitation learning for driving simulation. *arXiv preprint arXiv:1803.01044*.

[Bishop, Welch, and others 2001] Bishop, G.; Welch, G.; et al. 2001. An introduction to the kalman filter. *Proc of SIGGRAPH, Course* 8(27599-3175):59.

[Bobrowski et al. 2008] Bobrowski, O.; Meir, R.; Shoham, S.; and Eldar, Y. 2008. A neural network implementing optimal state estimation based on dynamic spike train decoding. In *Advances in Neural Information Processing Systems*, 145–152.

[Cho et al. 2014] Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Colyar and Halkias 2007] Colyar, J., and Halkias, J. 2007. Us highway 101 dataset. *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*.

[Coskun et al. 2017] Coskun, H.; Achilles, F.; DiPietro, R. S.; Navab, N.; and Tombari, F. 2017. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In *ICCV*, 5525–5533.

[Deng, Yu, and others 2014] Deng, L.; Yu, D.; et al. 2014. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing* 7(3–4):197–387.

[Deo and Trivedi 2018] Deo, N., and Trivedi, M. M. 2018. Convolutional social pooling for vehicle trajectory prediction. *arXiv preprint arXiv:1805.06771*.

[Graves et al. 2009] Graves, A.; Liwicki, M.; Fernández, S.; Bertolami, R.; Bunke, H.; and Schmidhuber, J. 2009. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence* 31(5):855–868.

[Graves, Mohamed, and Hinton 2013] Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, 6645–6649. IEEE.

[Haarnoja et al. 2016] Haarnoja, T.; Ajay, A.; Levine, S.; and Abbeel, P. 2016. Backprop kf: Learning discriminative deterministic state estimators. In *Advances in Neural Information Processing Systems*, 4376–4384.

[Helbing and Molnar 1995] Helbing, D., and Molnar, P. 1995. Social force model for pedestrian dynamics. *Physical review E* 51(5):4282.

[Helbing, Farkas, and Vicsek 2000] Helbing, D.; Farkas, I.; and Vicsek, T. 2000. Simulating dynamical features of escape panic. *Nature* 407(6803):487.

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

[Kalman 1960] Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering* 82(1):35–45.

[Krishnan, Shalit, and Sontag 2015] Krishnan, R. G.; Shalit, U.; and Sontag, D. 2015. Deep kalman filters. *arXiv preprint arXiv:1511.05121*.

[Kuefler et al. 2017] Kuefler, A.; Morton, J.; Wheeler, T.; and Kochenderfer, M. 2017. Imitating driver behavior with generative adversarial networks. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, 204–211. IEEE.

[Langford, Salakhutdinov, and Zhang 2009] Langford, J.; Salakhutdinov, R.; and Zhang, T. 2009. Learning nonlinear dynamic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 593–600. ACM.

[Lefèvre, Vasquez, and Laugier 2014] Lefèvre, S.; Vasquez, D.; and Laugier, C. 2014. A survey on motion prediction and risk assessment for intelligent vehicles. *Robomech Journal* 1(1):1.

[Lu and Skabardonis 2007] Lu, X.-Y., and Skabardonis, A. 2007. Freeway traffic shockwave analysis: exploring the ngsim trajectory data. In *86th Annual Meeting of the Transportation Research Board, Washington, DC*.

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

[Simon 2006] Simon, D. 2006. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons.

[Thrun, Burgard, and Fox 2005] Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic robotics*. MIT press.

[Vemula, Muelling, and Oh 2017] Vemula, A.; Muelling, K.; and Oh, J. 2017. Social Attention: Modeling Attention in Human Crowds. *ArXiv e-prints*.

[Wilson and Finkel 2009] Wilson, R., and Finkel, L. 2009. A neural implementation of the kalman filter. In *Advances in neural information processing systems*, 2062–2070.

Supplementary Materials

We offer a rough theoretical analysis of the correctness of SAKNN in this section. There is no concrete mathematical foundation to prove the results in DNN approach but we quantitatively analysis our algorithm and try to give an upper bound to show why SAKNN works. In the beginning, we formulate our algorithm in a mathematical language. Suppose the underlying acceleration and the measurement noise are denoted as a_i and $\xi_i \sim \mathcal{N}(0, \Sigma_i)$ from the time step 0 to the time step $N-1$ respectively, where Σ_i is the covariance matrix with respect to the time step i. Actually, we have only access to the sensor readers $a_i + \xi_i$ from the starting time 0 to the terminal time $N-1$.

We start our trajectory from the position p_0 with an observed velocity v and move to p_N after N time steps. According to the basic kinematic motion formula and our known sensor readers $a_i + \xi_i$, the accumulated position at the n-th time step p_n is written as

$$p_n = p_0 + n \cdot \Delta t \cdot v + \frac{\Delta t^2}{2} \cdot \sum_{i=1}^{n} (2i-1) \cdot (a_i + \xi_i) + \mathcal{O}\Big(p^{(3)}\Big),$$

where we use big $\mathcal{O}$ notation to simply represent the higher order term of Taylor's expansion of position. A simple calculation shows that the displacement error $\mathcal{E}_n$ between the underlying position and accumulated position at the n-th time step is

$$\mathcal{E}_n = \frac{\Delta t^2}{2} \cdot \sum_{i=1}^{n} (2i-1) \cdot \xi_i + \mathcal{O}\Big(p^{(3)}\Big).$$

Thus, the upper bound of the displacement error is

$$\frac{(n \cdot \Delta t)^2}{2} \cdot \max_i(|\xi_i|) + \mathcal{O}\Big(p^{(3)}\Big). \tag{2}$$

Why Does SAKNN Work?

The architecture of SAKNN has two neural network structures, i. e. the interaction layer and the Kalman layer with three merits robustness, effectiveness and smoothness. We use a mathematical way to describe the merits of SAKNN:

- **The interaction layer**: SAKNN predicts the future acceleration but in fact we learns the signum of acceleration in the fitting loss. We use the symbol $\mathbb{P}$ to represent the model accuracy of predicted signum.

  $$\mathbb{P} := \text{Prob(SAKNN predicts the right signum of acceleration)}$$

- **The Kalman layer**: An effect of the Kalman layer is to decrease the amplitude of the observed sensor acceleration which is credited to the deep structure of neural networks in the interaction layer and the recursive calculation in the kalman layer. We use M_i and K_i to represent the order of the decreasing for the underlying acceleration and the measurement noise at the i-th time step respectively

  $$\frac{1}{M_i} \cdot a_i + \frac{1}{K_i} \cdot \xi_i,$$

  where $1 < M_i << K_i$. Because SAKNN improves the signal-to-noise ratio of the predicted signal significantly. Furthermore, we assume the order of the decreasing is consistent due to the smoothness of SAKNN and thus we omit the subscripts of M and K. Specifically, we write $M := M(\mathcal{W}; b)$ and $K := K(\mathcal{W}; b)$, where $\mathcal{W}; b$ represent the neural network.

We compute the displacement error $\mathcal{E}_n$ of SAKNN in two cases. One case is that when SAKNN predicts the right signum of acceleration, we write the displacement error of acceleration $\mathcal{E}(a_i)$ as

$$\mathcal{E}(a_i) = [(1 - \frac{1}{M}) \cdot a_i - \frac{1}{K}\xi_i].$$

And, another case is when SAKNN predicts wrong, the displacement error is

$$\mathcal{E}(a_i) = [(1 + \frac{1}{M}) \cdot a_i + \frac{1}{K}\xi_i].$$

Let $\mathcal{A} \subset \{0, \cdots, N-1\}$ to be the subset of the time index in which SAKNN predicts the right signum of acceleration and $\mathcal{B} = \{0, \cdots, N-1\}/\mathcal{A}$ be the wrong one.

Then, the displacement error $\mathcal{E}_n$ of SAKNN at the n-th time step is

$$\frac{\Delta t^2}{2} \cdot \left(\mathbb{P} \cdot \sum_{i \in \mathcal{A}} (2i-1) \cdot [(1 - \frac{1}{M}) \cdot a_i - \frac{1}{K}\xi_i] + (1 - \mathbb{P}) \cdot \sum_{i \in \mathcal{B}} (2i-1) \cdot [(1 + \frac{1}{M}) \cdot a_i + \frac{1}{K}\xi_i] \right) + \mathcal{O}\Big(p^{(3)}\Big). \tag{3}$$

Observe that a well-turned SAKNN yields that the predicted probability approaches to 1, the predicted acceleration approaches to the underlying one, and the signal-to-noise ratio becomes bigger, i.e. $\mathbb{P} \longrightarrow 1$, $M \longrightarrow 1$ and $1 < M << K$. Then, the first term of the displacement error (3) becomes

$$\mathbb{P} \cdot \sum_{i \in \mathcal{A}} (2i-1) \cdot [(1 - \frac{1}{M}) \cdot a_i - \frac{1}{K} \xi_i] \longrightarrow \sum_{i \in \mathcal{A}} \frac{1-2i}{K} \cdot \xi_i \tag{4}$$

,and the second term becomes

$$(1 - \mathbb{P}) \cdot \sum_{i \in \mathcal{B}} (2i-1) \cdot [(1 + \frac{1}{M}) \cdot a_i + \frac{1}{K} \xi_i] \longrightarrow 0. \tag{5}$$

We plug (4) and (5) into (3). The displacement error $\mathcal{E}_n$ of a well-turned SAKNN becomes

$$\mathcal{E}_n(\text{SKANN}) = \frac{\Delta t^2}{2} \cdot \sum_{i \in \mathcal{A} = \{0, \cdots, N-1\}} \frac{1-2i}{K} \cdot \xi_i + \mathcal{O}\Big(p^{(3)}\Big).$$

Thus, the upper bound of the displacement error is

$$\frac{(n \cdot \Delta t)^2}{2} \cdot \frac{\max_i(|\xi_i|)}{K} + \mathcal{O}\Big(p^{(3)}\Big) \tag{6}$$

Comparing (2) and (6), since the parameter K is a relatively big number determined by the SAKNN and we can claim that the predicted trajectory of SAKNN has a very lower displacement error almost equally to the third order remainder of position. The experimental results confirm the great performance of SAKNN and our analysis in the end.