Probabilistic Estimation of Vehicle Speed for Autonomous Vehicles using Deep Kernel Learning

Esa Apriaskar*
School of Electrical and Electronic Engineering
University of Sheffield, United Kingdom
Department of Electrical Engineering
Universitas Negeri Semarang, Indonesia
eapriaskar1@sheffield.ac.uk

Xingyu Liu
School of Electrical and Electronic Engineering
University of Sheffield, United Kingdom
xliu231@sheffield.ac.uk

Amornyos Horprasert
School of Electrical and Electronic Engineering
University of Sheffield, United Kingdom
ahorprasert1@sheffield.ac.uk

Lyudmila Mihaylova
School of Electrical and Electronic Engineering
University of Sheffield, United Kingdom
l.s.mihaylova@sheffield.ac.uk

***Abstract*—Perception systems of autonomous vehicles (AVs) play a crucial role in achieving different levels of autonomy and interpreting information from complex traffic environments. The inherent uncertainties are a persistent factor. While some environmental features can be directly measured by certain sensors, measuring accurately the velocity of moving vehicles presents a substantial challenge. To this end, this paper demonstrates the utilisation of a powerful non-parametric method, Gaussian process regression, in combination with a method that leverages deep neural networks, known as Deep Kernel Learning (DKL), to estimate the vehicle speed using other existing data in the simulation that are considered feasible in real-world scenarios.**

The methodology is experimentally evaluated in a single ring-shaped traffic simulation where an autonomous vehicle (AV) drives together with human-driven vehicles (HDVs). The study reveals that the approach significantly enhances the accuracy and confidence of speed estimation with 64.58% and 50% improvements in the root mean square error (RMSE) for 525 and 3000 training data, respectively. It outperforms the conventional Gaussian processes, which suffer from a large dataset.

***Index Terms*—autonomous vehicle, Gaussian processes, speed estimation, deep kernel learning, SUMO**

I. Introduction

Autonomous vehicles (AVs), which have been extensively studied and developed for decades, have demonstrated promising potential in traffic smoothing. The control algorithms have evolved from conventional control methods, which aim to imitate the human driving behavior [1], to advance machine learning techniques such as reinforcement learning (RL) [2], demonstrating outstanding performance in improving the traffic flow across various traffic environments. Despite their differences, these algorithms share one common requirement: properly perceiving and processing significant environmental information to execute specific tasks. Modern methods, like those in [3]–[5], emphasise on the vehicle's acceleration, velocity and relative distance between the AV and surrounding vehicles, which were assumed to be directly measurable.

In real-world scenarios, distance can be easily obtained by proximity sensors such as ultrasonic sensors, LIDAR, etc. However, determining the velocity of an observed vehicle poses a significant challenge, as those sensors cannot directly measure it but must be inferred through vision-based algorithms similar to those used by stationary speed cameras on highways [6], which will be a completely different case since the AVs are non-stationary. The challenge is rarely addressed in the existing literature on AVs for traffic flow control, which constitutes the primary focus of this study. One study tried to incorporate real-time data from a service provider to control design, but it still encountered significant delays and noise in the data [7]. It is also worth noting that existing studies less consider uncertainty awareness, which could potentially be a critical aspect of RL control design [8].

We are interested in estimating a vehicle's velocity using existing information that is easily obtainable in real-world conditions, where uncertainty is inevitable. To address this challenge, Gaussian processes (GPs) [9], a powerful stochastic process used in machine learning that models the underlying function of a collection of data as a distribution over functions, can be an option. It allows probabilistic estimation, which predicts uncertainty through variance rather than only an explicit value. These capabilities make GPs well-suited for environments where uncertainty is a significant factor.

While GPs are powerful tools for modelling environments characterised by uncertainty [10], GPs face significant scalability challenges with large datasets [11]. To address this limitation, we aim to enhance the performance of GPs by integrating deep neural networks, particularly within the con-

The authors thank the Indonesia Endowment Fund for Education (LPDP) for the funding support during the PhD research of Esa Apriaskar with contract no. 20230722299644. We are also grateful to the support from the UK EPSRC through Project NSF-EPSRC: "ShiRAS: Towards Safe and Reliable Autonomy in Sensor Driven Systems", under Grant EP/T013265/1. ShiRAS was also supported by the USA National Science Foundation under Grant NSF ECCS 1903466.

*Corresponding author, ORCID = 0000-0002-2171-1958 text of recent advancements introduced in [12], *Deep Kernel Learning* (DKL). It offers the ability to addresses the challenge of scalability while maintaining the expressiveness of GPs.

In this study, we present a unified framework that combines Gaussian processes with deep learning to enhance uncertainty quantification. In contrast to [13] where the hyperparameters are learned by maximum likelihood estimation, this paper leverages hyperparameters learning with deep layers of neural networks. We adopt Deep Kernel Learning (DKL) and predict uncertainty intervals. These are then used to probabilistically estimate vehicle velocity in a simulation environment. We also compare the performance of conventional Gaussian processes as the baseline with the DKL approach, discussing their respective advantages and limitations.

The remainder of this article is organised as follows: Section II introduces the background knowledge of GPs, Section III details the proposed simulation environment and methods. Section IV discussed the results, and finally, Section V is for the conclusion and possible enhancement for future work.

II. Background Methodology

A. Gaussian Processes (GP)

GP methods are highly effective tools for modelling the dynamics of a system by offering probabilistic distributions over functions. The flexibility of GP methods in data-driven applications is attributed to their capability to represent complex relationships within data. This is achieved through the incorporation of uncertainty quantification using mean functions and covariance matrices. The modelling by GP methods makes them especially advantageous in the field of control design, where the consideration of uncertainty is of utmost importance.

The capability of GP methods in such applications relies on three essential components: prior, posterior, and hyperparameter optimisation. The prior distribution represents the initial beliefs about the function before any new data is given, whereas the posterior distribution updates these beliefs based on the newly observed data, resulting in improved estimations. Hyperparameter optimisation, on the other hand, involves adjusting the parameters that specify the covariance function in order to improve the model's performance and accuracy.

B. GP Prior and Posterior

In the context of predicting the dynamics of a system, the GP prior assumes that the system behaviour given a set of existing input can be distributed using a multivariate Gaussian distribution. It means that any collection of function values $f(x_1), f(x_2), ...f(x_n)$, which are the system behaviour given any collection of inputs $x_1, x_2, ...x_n$, has a joint multivariate Gaussian distribution as expressed as

$$p(X;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(\frac{(X-\mu)^\top \Sigma^{-1}(X-\mu)}{-2}\right), \tag{1}$$

where μ is the mean of the $f(X)$ value, which is commonly considered zero ($\mu_0 = 0$) for simplicity in GP prior with a normal distribution. Σ is a covariance matrix, denoted as n x n matrix, which has to be symmetric positive definite to be valid for a multivariate Gaussian distribution. To generate the covariance matrix Σ, a radial-basis-function (RBF) kernel

$$K_{\text{RBF}}(X,X) = K(x_i,x_j) = a^2 \exp\left(-\frac{1}{2l^2}\|x_i - x_j\|^2\right), \tag{2}$$

which is the most common function in GP methods. Thus, in the GP prior, $f(X) \sim \mathcal{N}(\mu_0, \Sigma_0)$ with Σ_0 can be calculated using (2) with the existing input x. The GP posterior connects the prior assumption on system behaviour $f(X)$ to our observation in the output of the system $y(X)$

$$y(X) = f(X) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{3}$$

with σ^2 being the variance of the Gaussian noise ϵ. Note that $y(X)$ is a vector for the observation output $(y(x_1), y(x_2), ...y(x_n))^\top$ and $f(X)$ is the vector for latent noise-free function values $(f(x_1), f(x_2), ...f(x_n))^\top$ as prior knowledge, which both are indexed by a set of inputs $X = (x_1, x_2, ...x_n)$. Since $f(X)$ and ϵ are both multivariate Gaussian distributions, $y(X)$ is considered as a sum of two independent multivariate Gaussian variables, making the distribution of $y(X) \sim \mathcal{N}(0, K_\sigma = K(X,X) + \sigma^2 I)$, where $K(X,X)$ is the RBF kernel function of the inputs X.

Considering a new input data $X_* = (x_{*1}, x_{*2}, ...x_{*m})$ that generates $f(X_*)$, therefore joint distribution over f_* and y is obtained as

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_\sigma & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{4}$$

From the joint distribution, a conditional distribution of the new function given the observation output and input data $f_*|y,X,X_*$ is obtained using the following equations:

$$f_*|y,X,X_* \sim \mathcal{N}(\mu_*, \Sigma_*), \tag{5}$$

$$\mu_* = K(X_*,X)K_\sigma^{-1}y, \tag{6}$$

$$\Sigma_* = K(X_*,X_*) - K(X_*,X)K_\sigma^{-1}K(X,X_*). \tag{7}$$

C. Hyperparameter Optimisation

The process of hyperparameter optimisation involves maximising the inference of the Gaussian distribution for the posterior. The objective of this part is to refine the hyperparameter of the covariance function in order to improve the accuracy of estimation. It is important to note that within the RBF kernel, there exist two hyperparameters $\theta = (a, l)$. In order to determine the most optimal hyperparameters (θ), one can employ a log marginal Gaussian likelihood function

$$\mathcal{L} = -\frac{1}{2}y^\top K_\sigma^{-1}y - \frac{1}{2}\log|K_\sigma| - \frac{n}{2}\log(2\pi). \tag{8}$$

Note that in the case where the noise variance σ^2 is unknown, it can be included as additional hyperparameter to optimise. Gradient-based optimisation methods, such as Adam [14], can be used to tune the hyperparameters.

III. Proposed Work

A. System Environment

This work focuses on estimating the speed of the vehicle in front of an AV in a single-ring road environment. The environment discussed is a single-lane circular road with a circumference of $C = 200$ metres with $N = 19$ vehicles denoted by $v_1, v_2, \ldots, v_N$, as illustrated in Fig. 1. The AV, coloured in red, and human-driven vehicles (HDVs), coloured in white and blue, are driving endlessly in the given road environment. The driving behaviour of all vehicles is based on the well-known car following model, the Intelligent Driver Model (IDM), as presented in [1], but they have different information. While HDVs only have information about their own speed and acceleration, the AV can estimate the leading vehicle speed using the method we propose in this paper. This additional information is used by the AV control system.

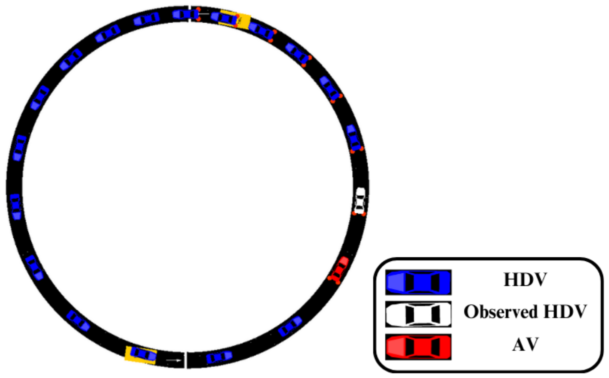


Fig. 1. Single-lane Ring Road Environment. The AV is colored in red, while the observed HDV is in white. The blue vehicles are unobserved HDVs.

The estimation in this work is performed with uncertainty awareness in the training data, preparing the leading vehicle speed (v_l) to have noise with variance σ^2. We assume that the methods do not have any information about the value of σ^2, but aim to follow the true value in the training process. Note that as one of the observed states which are not coming from the agent vehicle, v_l is beneficial in the control design for vehicular traffic systems [2], [4].

B. Deep Kernel Learning (DKL)

Combining neural networks with the GP approach, deep kernel learning (DKL) emerges the representational power and uncertainty quantification ability. While conventional GP methods may only justify the confidence of predictions given noise and uncertainty, neural networks enable DKL to learn more complex non-linear dynamics. The DKL structure is divided into two parts: the neural network and Gaussian process parts as depicted in Fig. 2. The parametric multi-layer perceptron in the hidden layers that follow the input layer reflects the neural network part. The Gaussian process part is the following hidden layer that has an infinite number of basis functions with base kernel hyperparameters (θ), also known as the Kernel layer.

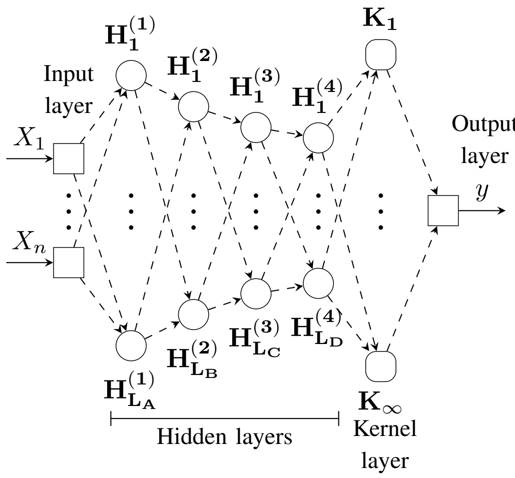


Fig. 2. The structure of DKL model. Four hidden layers with 1000-500-50-8 nodes and hyperbolic tangent activation function are utilised prior to the kernel layer, which generates the output prediction.

The DKL method can be described as a base kernel function, denoted as $K(X, X)$, which incorporates hyperparameters (θ). Nevertheless, the application of a non-linear mapping from the neural network part $g(X, \mathbf{w})$ alters the inputs of this particular function as demonstrated:

$$K(X, X|\theta) \rightarrow K(g(X, \mathbf{w}), g(X, \mathbf{w})|\theta, \mathbf{w}), \tag{9}$$

where X represents the input of the model, while $\mathbf{w}$ denotes the hyperparameter of the neural network part. The network is provided with a [4 x 1]-column vector

$$X = \left[a_e, v_e, d_l, \Delta_{d_l}\right]^\top ; y = v_l \tag{10}$$

as input, which represents the AV's acceleration (a_e), speed (v_e), leader distance (d_l), and the rate of change of leader distance (Δ_{d_l}). The leader distance, denoted as d_l, represents the spatial separation between the AV and the observed HDV, referred to as the leading vehicle. The output of the model (y) in this context is the velocity of the leading vehicle v_l.

The RBF kernel, as expressed in (2), is widely employed as the base kernel function. However, Wilson et al. [15] argue that a spectral mixture base kernel

$$K_{SM}(X, X|\theta) = \sum_{q=1}^{Q} a_q \left|\Sigma_q\right|^{\frac{1}{2}} (2\pi)^{-D/2}$$
$$\exp\left(-\frac{1}{2}\left\|\Sigma_q^{\frac{1}{2}}\left(x_i - x_j\right)\right\|^2\right) \cos\langle x_i - x_j, 2\pi\mu_q\rangle \tag{11}$$

provides greater flexibility than a single kernel only in capturing intricate relationships and patterns. Here, Q denotes the number of special mixture components and hyperparameters of the kernel $\theta = \{a_q, \Sigma_q, \mu_q\}$ are mixture weights, bandwidths, and frequencies, respectively. In this work, all hyperparameters $\gamma = \{\theta, \mathbf{w}\}$ are jointly optimised by maximising the log marginal likelihood $\mathcal{L}$ as shown in equation (8), using Adam optimizer [14].

C. Kernel Approximation

A key variable to take into account when incorporating DKL and GP methods in this study is the computation of the kernel matrix. As both methods are categorised as supervised learning, prior training is necessary. The nature is that higher number of the training data makes the training result better, but consequently, the dimension of the kernel matrix is higher, making the training runtime longer. To fasten the runtime for large training data, a kernel approximation is conducted using Lanczos algorithm [16].

Algorithm 1 Lanczos algorithm for kernel approximation

```
 1: Input: Exact kernel matrix $K_\sigma$
 2: Initialize:
 3:     $k \leftarrow 100$                                   ▷ Desired rank
 4:     $\mathbf{v}_1 \in \mathbb{R}^n$                      ▷ Initial vector
 5:     $\mathbf{v}_1$: $\mathbf{v}_1 = \mathbf{v}_1/\|\mathbf{v}_1\|$    ▷ Normalization
 6: for $t = 1$ to $k$ do                                ▷ Next vector computation
 7:     $\mathbf{u} = K_\sigma \mathbf{v}_t - \beta_{t-1}\mathbf{v}_{t-1}$    ▷ Set $\beta_{t-1} = 0$ for $t = 1$
 8:     $\alpha_t = \mathbf{v}_t^\top \mathbf{u}$
 9:     $\mathbf{u} = \mathbf{u} - \alpha_t \mathbf{v}_t$     ▷ Update $\mathbf{u}$
10:     $\beta_t = \|\mathbf{u}\|$
11:     $\mathbf{v}_{t+1} = \mathbf{u}/\beta_t$              ▷ Next vector
12: end for
13: $T = \text{tridiagonal}(\beta_{1:k-1}, \alpha_{1:k}, \beta_{1:k-1})$    ▷ Matrix $T$
14: $V = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k]$    ▷ Vector $V$
15: $\widetilde{K_\sigma} \approx VTV^\top$              ▷ $\widetilde{K_\sigma}$ approximation
16: Output: Approximated kernel matrix $\widetilde{K_\sigma}$
```

D. Log-likelihood Approximation

The possibility of numerically unstable matrix calculation in the kernel layer is another essential consideration. It can result from the computation in (8). Numerical instability of the matrix may arise when inverse matrix is calculated directly and the log determinant [9] can be used in such cases. A poor or ill-conditioned kernel matrix that is challenging to invert as a result of searching the optimal kernel hyperparameters may be the source of this instability, which could result in an imprecise or undefinable variance prediction.

During the hyperparameter optimisation, the use of gradient-based method requires the gradient of the log marginal likelihood

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{2} y^\top K_\sigma^{-1} \frac{\partial K_\sigma}{\partial \theta} K_\sigma^{-1} y - \frac{1}{2}\,\text{Tr}\left(K_\sigma^{-1}\frac{\partial K_\sigma}{\partial \theta}\right) \tag{12}$$

which urges the approximation of Tr $\left(K_\sigma^{-1}\frac{\partial K_\sigma}{\partial \theta}\right)$ for the same reason. All three terms are obtained using the modified batch conjugate gradients (mBCG) technique described in [17], in order to circumvent the conditions. Note that V_i is obtained with Lanczos algorithm and probe vector z_i. With this method, large matrices can also be calculated more quickly using this method than with the widely used Cholesky decomposition.

IV. Results and Discussions

We conducted two different cases for leading vehicle speed estimation in single ring-road environment using: 1) 105-seconds data (equivalent to $n = 525$ data points), and 2) 600-seconds data (equivalent to $n = 3000$ data points) for training,

Algorithm 2 mBCG algorithm for log-likelihood approximation

```
 1: Input: Approximated Kernel matrix $\widetilde{K_\sigma}$, observation $y$
 2: Initialize:
 3:     $t \leftarrow 10$                        ▷ The number of random vector $z_i$
 4:     $k_c \leftarrow 100$                     ▷ Maximum iterations
 5: for $i = 1$ to $t$ do
 6:     $z_i \sim \mathcal{N}(0, I)$             ▷ Random vectors
 7: end for
 8: $S_0 \leftarrow K_\sigma^{-1}[y, z_1, \dots, z_t] = [0, 0, \dots, 0]$    ▷ Initial guess
 9: $R_0 \leftarrow [y, z_1, \dots, z_t]$       ▷ Initial residuals
10: $P_0 \leftarrow R_0$                        ▷ Initial search directions
11: for $j = 0$ to $k_c - 1$ do
12:     $V_j \leftarrow \widetilde{K_\sigma} P_j$
13:     $\alpha_j^c \leftarrow \frac{(R_j \circ R_j)^\top \mathbf{1}}{(P_j \circ V_j)^\top \mathbf{1}}$    ▷ Step size for solutions, residuals
14:     $S_{j+1} \leftarrow S_j + \alpha_j^c P_j$    ▷ Update solution vectors
15:     $R_{j+1} \leftarrow R_j - \alpha_j^c V_j$    ▷ Update residual vectors
16:     $\beta_j^c \leftarrow \frac{(R_{j+1} \circ R_{j+1})^\top \mathbf{1}}{(R_j \circ R_j)^\top \mathbf{1}}$    ▷ Step size for search directions
17:     $P_{j+1} \leftarrow R_{j+1} + \beta_j^c P_j$    ▷ Update search directions
18: end for
19: $T_i = \text{tridiagonal}\left(\left\{\frac{\sqrt{[\beta_1^c]_i}}{[\alpha_1^c]_i}, \frac{\sqrt{[\beta_2^c]_i}}{[\alpha_2^c]_i}, \dots, \frac{\sqrt{[\beta_{k-1}^c]_i}}{[\alpha_{k-1}^c]_i}\right\},\right.$
20:     $\left\{\frac{1}{[\alpha_1^c]_i}, \frac{1}{[\alpha_2^c]_i} + \frac{[\beta_1^c]_i}{[\alpha_1^c]_i}, \dots, \frac{1}{[\alpha_k^c]_i} + \frac{[\beta_{k-1}^c]_i}{[\alpha_{k-1}^c]_i}\right\},$
21:     $\left.\left\{\frac{\sqrt{[\beta_1^c]_i}}{[\alpha_1^c]_i}, \frac{\sqrt{[\beta_2^c]_i}}{[\alpha_2^c]_i}, \dots, \frac{\sqrt{[\beta_{k-1}^c]_i}}{[\alpha_{k-1}^c]_i}\right\}\right)$    ▷ Matrix $T_i$
22: $\log|K_\sigma| \approx \sum_{i=1}^{t} z_i^\top V_i (\log T_i) V_i^\top z_i$    ▷ log determinant term
23: Tr $\left(K_\sigma^{-1}\frac{\partial K_\sigma}{\partial \theta}\right) \approx \frac{1}{t}\sum_{i=1}^{t} z_i^\top \left(K_\sigma^{-1}\frac{\partial K_\sigma}{\partial \theta}\right) z_i$    ▷ trace term
24: Output: approximated $K_\sigma^{-1} y$, $\log|K_\sigma|$, and Tr $\left(K_\sigma^{-1}\frac{\partial K_\sigma}{\partial \theta}\right)$
```

which challenge the method to work with a relatively different size of training data. All the training data are obtained from a microscopic traffic simulator, SUMO (Simulation of Urban MObility) [18]. We set the baseline for both cases with a standard GP methods that uses Lanczos and mBCG algorithms for kernel and log-likelihood approximation. To generate the kernel matrix, we utilise spectral mixture Kernel with $Q = 4$. The parameters setting of DKL and GP methods in both cases, along with the environment parameters, are shown in Table I.

TABLE I
Methods and Environment Parameters

Parameters	Value
Q number of mixtures in SM kernel	4
k rank size of approximated kernel matrix	100
t number of vectors for trace estimation	10
k_c maximum iteration of mBCG	100
η learning rate Adam optimizer	0.001
β_1 first-moment Adam optimizer	0.9
β_2 second-moment Adam optimizer	0.999
n_e training epochs	1000
σ standard deviation noise	0.5
C circumference of ring road (m)	200
N number of vehicles	19
δ simulation time-step (s)	0.2

Fig. 3 shows the performance comparison of DKL and the baseline with a similar number of training epochs ($n_e = 1000$)

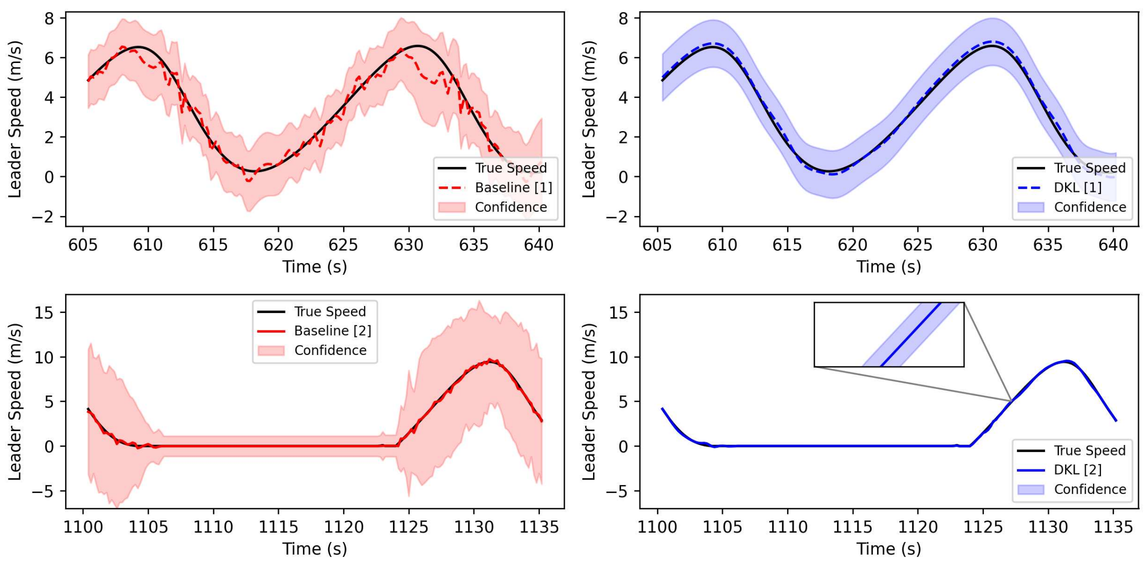
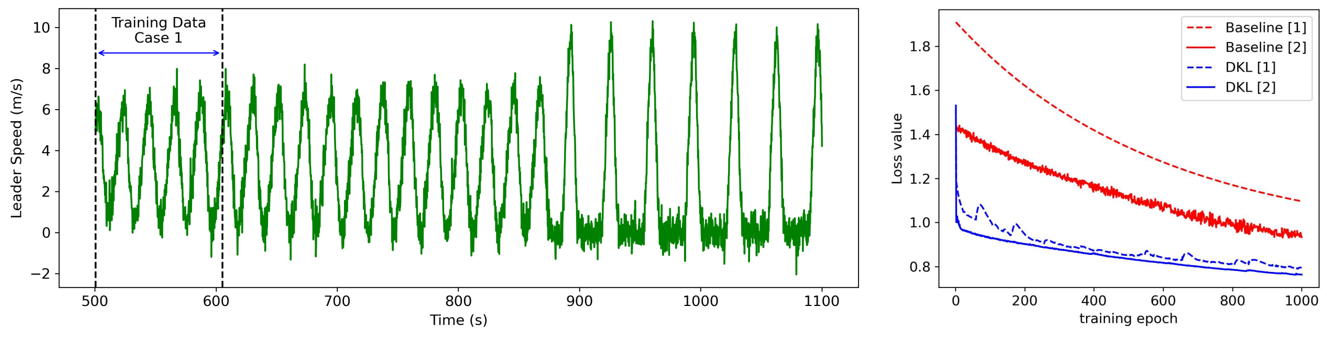


Fig. 3. Performance comparison between DKL and baseline methods for leading vehicle speed estimation. **Upper** (*Left to Right*): Baseline and DKL methods for the first case. **Lower** (*Left to Right*): Baseline and DKL methods for the second case. The confidence bound represents 95% confidence interval or equivalent to a 2σ band.



(a)

(b)

Fig. 4. (a) Noisy velocity of the leading vehicle (v_l) with $\sigma = 0.5$ for training data. The first case only takes the first 105 seconds of the data for training, while the second case takes the whole 600 seconds. (b) Loss value progression over training epochs between DKL and GP methods. The loss value is calculated from negative log marginal likelihood.

and same optimisation procedure for both scenarios. The two methods are employed to estimate the velocity of the leading vehicle over a duration of 35 seconds. The figures clearly demonstrate that in the first scenario, the DKL performance surpasses the baseline. Despite the fact that the confidence bound, established with a 95% confidence interval, covers the true value, the projected average of the baseline exhibits a fluctuating trajectory, suggesting a poor performance.

The maximum deviation occurs during the transition from acceleration to deceleration, potentially associated with the presence of significant noise in the training data for this specific transition (as depicted in Fig. 4a at approximately 570 seconds). The DKL estimation also indicates a deviation in the same transition, although it is not statistically significant when compared to the baseline.

In the second scenario, when the two methods are trained using a larger dataset, the training data also exhibits a changing trend in the stop-and-go speed, as depicted in Fig. 4a. This pattern increases the complexity of the data. Fig. 3 also demonstrates that both methods exhibit a significantly accurate mean estimation for the second case, although some minor oscillations are present in the baseline method. Given increasing training data size, the DKL method exhibits a significantly reduced confidence bound. This suggests that the predicted mean value is highly reliable and solid. In contrast, the baseline approach has a high confidence bound, indicating poor uncertainty quantification, despite the predicted mean being close to the actual value. It suggests that the baseline method encounters difficulties in capturing the evolving pattern, resulting in a significant level of uncertainty.

An important factor contributing to the observed performance of baseline and DKL algorithms is due to the progressive decreasing value of the loss function during training iterations. Fig. 4b illustrates that the baseline exhibits a higher loss value at the end of the iteration in comparison with DKL. Additionally, based on the convergence of the loss value, DKL shows more reliable convergence with 1000 epochs than the baseline. The loss value in this context refers to the negative log marginal likelihood, specifically the negative value of 8. This implies that a smaller value indicates a more optimal hyperparameter solution for the prediction methods.

Fig. 4b also illustrates that the progression of loss values for both methods improves as the size of the training data increases. The final iteration achieves a smaller loss value compared to the first case, which had a smaller number of training data points. Furthermore, it is observed that DKL consistently exhibits superior performance, as evidenced by its smaller loss value. This suggests that DKL is able to provide well optimised solutions for the hyperparameters. As evident from Table II, the root mean square error (RMSE) value of the DKL prediction is consistently lower than that of the generic GP algorithm. Nevertheless, one drawback of utilising DKL is the increased computational time required. The DKL algorithm consistently allocates more time to training and exhibits a substantial increase in performance when provided with additional training data points.

TABLE II
COMPARISON OF RMSE AND TRAINING RUNTIME

n	**RMSE**		**Training runtime (s)**	
	DKL	**GP**	**DKL**	**GP**
525 (Case 1)	0.17	0.48	70.86	66.20
3000 (Case 2)	0.07	0.14	3992.02	3071.87

V. CONCLUSIONS

This study presents a probabilistic approach to estimating the speed of the leading vehicle by employing DKL. This method is designed to address the challenges posed by uncertain observations and scalability concerns. The results demonstrate that our trained model successfully improves the estimation's accuracy above the baseline. With the inclusion of larger datasets and the growing complexity of the training data, our model has the potential to significantly enhance its accuracy, while the baseline model appears to encounter difficulties with optimising the confidence bound.

The DKL method offers the advantage of effectively modelling complex and nonlinear systems while also providing uncertainty quantification with scalability concern. Nevertheless, it requires extensive training time. Possible future research directions may involve the incorporation of inducing points or other approximations within the kernel layer as a means to mitigate the runtime issue. An enhancement to the network architecture can also be developed to improve efficiency.

REFERENCES

[1] R. Herman, E. W. Montroll, R. B. Potts, and R. W. Rothery, "Traffic Dynamics: Analysis of Stability in Car Following," *Operations Research*, vol. 7, no. 1, pp. 86–106, 1959. Publisher: INFORMS.

[2] Z. Yan, A. R. Kreidieh, E. Vinitsky, A. M. Bayen, and C. Wu, "Unified Automatic Control of Vehicular Systems With Reinforcement Learning," *IEEE Transactions on Automation Science and Engineering*, vol. 20, pp. 789–804, Apr. 2023.

[3] C. Wu, A. R. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: A Modular Learning Framework for Mixed Autonomy Traffic," *IEEE Transactions on Robotics*, vol. 38, pp. 1270–1286, Apr. 2022.

[4] A. R. Kreidieh, C. Wu, and A. M. Bayen, "Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning," in *Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1475–1480, Nov. 2018. ISSN: 2153-0017.

[5] K. Jang, E. Vinitsky, B. Chalaki, B. Remer, L. Beaver, A. A. Malikopoulos, and A. Bayen, "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '19, (New York, NY, USA), pp. 291–300, Association for Computing Machinery, Apr. 2019.

[6] I. Nishitani, H. Yang, R. Guo, S. Keshavamurthy, and K. Oguchi, "Deep Merging: Vehicle Merging Controller Based on Deep Reinforcement Learning with Embedding Network," in *Proceedings of 2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 216–221, May 2020. ISSN: 2577-087X.

[7] N. Lichtlé, K. Jang, A. Shah, E. Vinitsky, J. W. Lee, and A. M. Bayen, "Traffic Smoothing Controllers for Autonomous Vehicles Using Deep Reinforcement Learning and Real-World Trajectory Data," in *Proceedings of 26th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 4346–4351, Sept. 2023. ISSN: 2153-0017.

[8] C. Bogoclu, R. Vosshall, K. Cremanns, and D. Roos, "Deep Gaussian Covariance Network with Trajectory Sampling for Data-Efficient Policy Search," in *Proceedings of 2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, pp. 1–7, Feb. 2024.

[9] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[10] T. Beckers and S. Hirche, "Prediction With Approximated Gaussian Process Dynamical Models," *IEEE Transactions on Automatic Control*, vol. 67, pp. 6460–6473, Dec. 2022.

[11] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian Process Meets Big Data: A Review of Scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, pp. 4405–4423, Nov. 2020.

[12] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep Kernel Learning," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 370–378, PMLR, May 2016. ISSN: 1938-7228.

[13] X. Liu, L. Mihaylova, J. George, and T. Pham, "Gaussian process upper confidence bounds in distributed point target tracking over wireless sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 1, pp. 295–310, 2022. Publisher: IEEE.

[14] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

[15] A. Wilson and R. Adams, "Gaussian Process Kernels for Pattern Discovery and Extrapolation," in *Proceedings of the 30th International Conference on Machine Learning*, pp. 1067–1075, PMLR, May 2013. ISSN: 1938-7228.

[16] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *Journal of Research of the National Bureau of Standards*, vol. 45, p. 255, Oct. 1950.

[17] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "GPyTorch: blackbox matrix-matrix Gaussian process inference with GPU acceleration," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, (Red Hook, NY, USA), pp. 7587–7597, Curran Associates Inc., Dec. 2018.

[18] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic Traffic Simulation using SUMO," in *Proc. of the 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2575–2582, Nov. 2018. ISSN: 2153-0017.