Proceedings Paper:
Zhu, Y., Wang, P. and Mihaylova, L. orcid.org/0000-0001-5856-2223 (2021) A convolutional neural network combined with a Gaussian process for speed prediction in traffic networks. In: 2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). IEEE International Conference on Multisensor Fusion and Integration (MFI 2021), 23-25 Sep 2021, Karlsruhe, Germany (online). Institute of Electrical and Electronics Engineers (IEEE) . ISBN 9781665445221

https://doi.org/10.1109/MFI52462.2021.9591204

A Convolutional Neural Network Combined with a Gaussian Process for Speed Prediction in Traffic Networks

Yifei Zhu[1], Peng Wang[2], and Lyudmila Mihaylova[1]

[1]Department of Automatic Control and Systems Engineering, University of Sheffield, S1 3JD, UK
Email: {yzhu42, l.s.mihaylova}@sheffield.ac.uk
[2]Manchester Metropolitan University, Manchester, United Kingdom, Email: p.wang@mmu.ac.uk

***Abstract*—This paper proposes a traffic speed prediction framework combining a Convolutional Neural Network (CNN) with a Gaussian Process (GP) and is an extension of ConvNet-GP [1]. The main focus is on spatio-temporal large scale traffic networks and on uncertainty quantification. The emphasis is on the impact on the measurement noises on the predicted traffic speeds. The Gaussian Process regression provides a variance which characterises the accuracy of the prediction. The traffic speed data is converted into a three dimensional format like images and these are inputs of the CNN-GP framework for traffic networks. The CNN-GP framework provides 18.23% average improvement of the speed root mean square error compared with the generic CNN and gives a quantitative characterisation of the noise effects.**

***Index Terms*—Traffic Prediction, Gaussian Process, Deep Neural Network, Large Scale, Scalability**

I. INTRODUCTION

Traffic congestion has become a problematic issue for most metropolitan areas and has drawn extensive attention in the light of traffic prediction and control systems. Model-based and data-driven methods [2] are the most popular ones. While model-based methods focus efforts on building representative and scalable physical models with different levels of detail about traffic networks [2], [3], data-driven methods mostly require only historical data. For instance, convolutional neural networks (CNNs) have been successfully implemented to understand spatio-temperal features of traffic data [4],[5]. Ma et al. [6] propose a novel approach which converts the traffic speed into images, where pixel intensities represent the speed, and the horizontal and vertical coordination represent road segments and timestamps, respectively. However, the adjacency in the image is not equivalent to the adjacency in the traffic network. In the case of applying CNNs to traffic systems, some spatial information will be neglected during the pooling operation [7]. Therefore, the capsule network (CapsNet) [8], [9] is utilised in traffic prediction to replace the pooling operation with dynamic routing. Other deep learning methods are under investigating as well, such as recurrent neural networks (RNNs) proposed to infer the interaction between adjacent road segments and time dynamics [10].

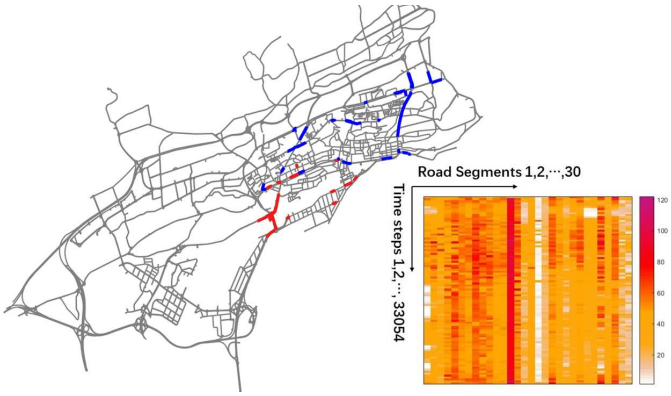

Figure 1. Road network of central Santander city. Red lines denote road segments where the speed sensors are located. On the rgiht, it is the image presentation of the traffic speed data, where x-axis represents locations and y-axis represents time. The pixel values represent the traffic speed.

Nevertheless, deep learning methods still face the challenges like time consuming and computational complexity. In addition, traditional deep learning models provide results without quantifying the impact of different uncertainties on the solutions. The knowledge of the impact of uncertainty on the results is particularly crucial in high-risk applications [11], [12]. Bayesian inference methods have successfully demonstrated learning capabilities in the presence of uncertainty in the data, in the models, prior information and other factors [13]. Gaussian Process (GP) methods are Bayesian representatives and have proven their power and their potential to equip CNNs with the ability of uncertainty analysis. Successful applications of GP methods to traffic prediction can be found in [14], [15]. In [16], a Gaussian process classifier is applied on the last-layer features of a CNN and achieves a performance similar to the performance of a CNN applied on the MNIST dataset [17]. Garriga et al. [1] propose a combination of a deep CNN and a shallow Gaussian process which has not only achieved surprisingly improved accuracy on the MNIST dataset but also enabled the CNN to be capable of uncertainty analysis. Inspired by this work, we have extended the idea of [1] on traffic speed forecasting, and we have shown its effectiveness by comparing it with typical CNNs and CNN with a GP

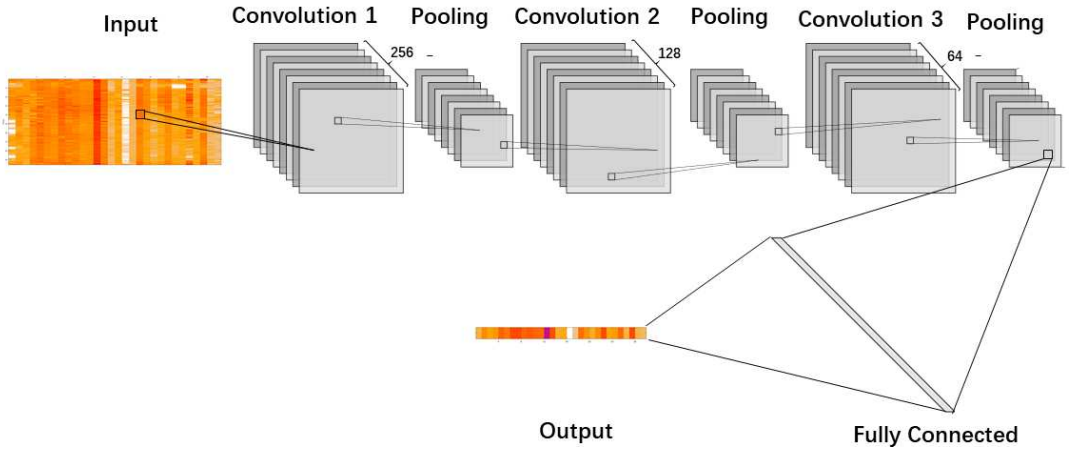

Figure 2. The typical CNN architecture. [6]

regression. As far as we know, this is the first application of a convolutional GP to a time series traffic prediction problem.

The main contributions of this paper are the following:

- A CNN-GP framework for traffic speed prediction in traffic networks is proposed. The CNN provides feature maps to the GP regression which predicts speeds and provides the variance that is used to quantify the impact of uncertainties;
- A deep CNN converted to a shallow GP [1], termed ConvNet GP, is implemented for the traffic speed prediction;
- A generic CNN is also implemented as a baseline. The CNN-GP and ConvNet GP are compared with the baseline CNN;
- A detailed performance validation and evaluation of these three approaches is conducted on real traffic data.

The remaining part of the paper is organised as follows. Section II. describes how the time series data from magnetic loop sensors are converted into an image format. Section II.B presents the structure of a generic CNN and of a CNN followed by a GP for prediction. Section II.C describes ConvNet GP. Section III provides performance evaluation. Section IV presents conclusions and summarises the results.

II. TRAFFIC SPEED PREDICTION

A. Traffic Speed Data as an Image

Speed detection sensors are installed on road segments in the traffic network. Figure 1 shows the sensor location in red. The sensor speed data are averaged in a certain time interval. For reserving the spatio-temporal feature of speed data, they are converted into images with two axes representing timestamp and location. Here X_{MN} represents the traffic speed value at the H-th time step on the D-th road segment.

$$\mathbf{X} = \begin{bmatrix} X_{11} & ... & X_{1D} \\ ... & ... & ... \\ X_{H1} & ... & X_{HD} \end{bmatrix} \tag{1}$$

B. CNN and CNN with GP Regression

This CNN has been proved to be significantly powerful not only in image processing but also in time series forecasting. Therefore, we take the CNN architecture proposed by [6] as one of the baselines. This architecture contains three convolutional layers, and a pooling layer follows each convolutional layer. A flattening and fully connected layer follow three pairs of convolutional layer and pooling layer. The detailed architecture is presented in [6]. The parameters are listed in Table I.

Proposed in [16], a Gaussian process can be used in conjunction with a CNN to analysis uncertainty without decreasing performance. Adding a GP regression in the last-layer features of the CNN will provide a tool to quantify the impact of the uncertainties on the classification process. In particular, the GP variance will be used as a measure of quantifying the results.

A Gaussian process is defined as a collection of random variables, with any finite number of which have a joint Gaussian distribution [18]. A Gaussian process, $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, is defined by mean function, $m(\mathbf{x})$, and covariance function, $k(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are the last-layer features obtained from the typical CNN. The regression problem can be solved by Bayesian inference with a GP prior. Let $\mathbf{f}$ be the observed function values for the training set, and let $\mathbf{f}_t$ be the set of function values corresponding to the testing set $\mathbf{x}_t$. Therefore, the joint distribution is defined as,

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_t \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_{xtx} \\ \mathbf{K}_{xtx}^T & \mathbf{K}_{xtxt} \end{bmatrix} \right), \tag{2}$$

where $\boldsymbol{\mu}_t$ is the test mean, $\mathbf{K}_{xtx}$ is the train-test covariance, and $\mathbf{K}_{xtxt}$ is the test-test covariance. Therefore, corresponding to the conditioning the joint Gaussian prior distribution on the observations is given by

$$\mathbf{f}_t|\mathbf{X}_t, \mathbf{X}, \mathbf{f} \sim \mathcal{N}(\mathbf{K}_{xtx}\mathbf{K}^{-1}\mathbf{f}, \mathbf{K}_{xtxt} - \mathbf{K}_{xtx}\mathbf{K}^{-1}\mathbf{K}_{xtx}^T). \tag{3}$$

Assuming additive independent identically distributed Gaussian noise with variance σ_n^2, the prior of the noisy obervation

becomes $cov(\mathbf{y}) = \mathbf{K} + \sigma_n^2\mathbf{I}$, where y is the ground truth observations. Following equation (4), the joint distribution of the observed values and the function values for test set can be written as,

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_t\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{K}+\sigma_n^2\mathbf{I} & \mathbf{K}_{xtx}\\ \mathbf{K}_{xtx}^T & \mathbf{K}_{xtxt}\end{bmatrix}\right). \tag{4}$$

The conditional distribution is, therefore, $f_t|\mathbf{X}, \mathbf{y}, \mathbf{X}_t \sim \mathcal{N}(\mathbf{K}_{xtx}[\mathbf{K}+\sigma_n^2\mathbf{I}]^{-1}\mathbf{f}, \mathbf{K}_{xtxt} - \mathbf{K}_{xtx}[\mathbf{K}+\sigma_n^2\mathbf{I}]^{-1}\mathbf{K}_{xtx}^T)$. The predictive mean and covriance functions are typically parameterised in therms of hyperparameters θ. During trainging, the value of the hyperparameters are defined by optimising the marginal log-likelihood:

$$L = \log p(\mathbf{y}|\mathbf{x},\theta) = -\frac{1}{2}|\mathbf{K}| - \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T\mathbf{K}^{-1}(\mathbf{y}-\boldsymbol{\mu}) - \frac{n}{2}\log(2\pi). \tag{5}$$

C. Deep CNN as a Shallow Gaussian Process

A deep neural network can be considered conceptually as a Gaussian process regression [19]. Hence, here we are adopting this vision and a deep CNN which can be considered as a shallow GP [1]. As pointed out in [20] combining the non-parametric nature of GP regression and learning ability of neural networks can improve the generalisation capabilities of the GPs. Details about the main ideas behind the ConvNet GP are given below.

1) Concepts and Definitions: As mentioned previously, CNNs' inaccurate uncertainty estimation is becoming increasingly problematic. For a standard CNN, with L hidden layers, the transformation from layer l to layer $l+1$ is given as following.

$$\mathbf{a}_j^{(l+1)}(\mathbf{V}) = \mathbf{b}_j^l + \sum_{i=1}^{C^l} \mathbf{W}_{j,i}^l \phi(\mathbf{a}_i^l(\mathbf{V})) \tag{6}$$

where

$$\mathbf{V} = \begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \cdot\\ \cdot\\ \cdot\\ \mathbf{v}_{C^0}\end{bmatrix} \tag{7}$$

is the input image with height H^0 and width D^0. Each input image has C^0 channels, and hence, the input image is considered as a $C^0 \times (H^0D^0)$ matrix, with $\mathbf{v}_i$ a row vector of size $1 \times (H^0D^0)$. The i-th output from the l-th layer is represented by $\mathbf{a}_i^{l+1}(\mathbf{V})$. The bias is $\mathbf{b}_j^l$ and the weight matrix that derives from the filter $\mathbf{U}_{j,i}^l$ at the l-th layer is $\mathbf{W}_{j,i}^l$. The activation for the output of the previous layer is represented as $\phi(\mathbf{a}_i^l(\mathbf{V}))$. For the first layer, $\phi(\cdot)$ just maps the input $\mathbf{v}_i$ to itself. In equation (6), $\mathbf{W}_{j,i}^l\phi(\mathbf{a}_i^l(\mathbf{V}))$ indicates the dot product of $\mathbf{W}_{j,i}^l$ and $\phi(\mathbf{a}_i^l(\mathbf{V}))$.

In the traditional CNN paradigm, the elements of a convolution filter $\mathbf{U}_{j,i}^l$ are determined. One filter is usually responsible for a specific feature. It is hard to say if the filter can still manage to capture the features when the data is polluted by random noise. To tackle the problem, one intuitive idea is to make the filter itself random rather than determined. In this way, for one specific convolution filter $\mathbf{U}_{j,i}^l$, when the elements change randomly, the number of potential filters could approach infinity. Therefore, all the filters together should be able to average out the noise and extract the features from the polluted data as described in [1]. A convolution filter $\mathbf{U}_{j,i}^l$ is substantially a matrix as shown in Fig. 3. As in [1], each element $\mathbf{u}_{j,i,x,y}^l$ of $\mathbf{U}_{j,i}^l$ is governed by a Gaussian distribution as in equation (8), and the bias $\mathbf{b}_j^l$ is governed by another Gaussian distribution as in equation (9).

$$u_{j,i,x,y}^l \sim \mathcal{N}(0, \frac{\sigma_w^2}{C^l}) \tag{8}$$

where i and j are the channel index in layer $l-1$ and l, x and y are the horizontal and vertical location of the element in the filter.

$$b_j^l \sim \mathcal{N}(0, \sigma_b^2) \tag{9}$$

As each single element $\mathbf{u}_{j,i,x,y}^l$ of $\mathbf{U}_{j,i}^l$ subjects to a Gaussian distribution, we can therefore derive as many filters as possible by sampling from the corresponding distribution. redEach filter can be further flattened into a weight matrix $\mathbf{W}_{j,i}^l$ with the dimension of $N^l \times (H^lD^l)$. According to equation (6), if $N^l \to \infty$, then $C^l \to \infty$. With the Central Limit Theorem (CLT), equation (8) and (9), we know that $\mathbf{a}_j^{l+1}(\mathbf{V})$ subjects to a Gaussian distribution as $C^l \to \infty$.

2) Mean and Covariance: In [1], the authors show that with certain constraints, a deep convolutional neural network is equivalent to a shallow Gaussian process. When it comes to a Gaussian process, we need to investigate how to obtain the mean and covariance from a deep convolutional neural network.

According to equation (6), we have one input image $\mathbf{V}$. To derive the covariance, we need at least another input image denoted as $\mathbf{V}'$. Indexed by $\mathbf{V}$ and $\mathbf{V}'$, we can now get two feature maps $\mathbf{a}_j^l(\mathbf{V})$ and $\mathbf{a}_j^l(\mathbf{V}')$ that can be concatenated as

$$\mathbf{a}_j^l(\mathbf{V}, \mathbf{V}') = (\mathbf{a}_j^l(\mathbf{V}), \mathbf{a}_j^l(\mathbf{V}'))^T, \tag{10}$$

which can be further extended as equation (11) to show the transformation from layer l to layer $l+1$,

$$\mathbf{a}_j^{l+1}(\mathbf{V}, \mathbf{V}') = \mathbf{b}_j^l\mathbf{I} + \sum_{i=1}^{C^l} \begin{bmatrix}\mathbf{W}_{j,i}^l & \mathbf{0}\\ \mathbf{0} & \mathbf{W}_{j,i}^l\end{bmatrix} \phi(\mathbf{a}_i^l(\mathbf{V}, \mathbf{V}')), \tag{11}$$

which is proven to be multivariate Gaussian in [1] when $C^l \to \infty$. In equation (11), $\mathbf{I}$ is the identity matrix. The attribute holds for any given feature map determined by j and l, which are further constrained by the structure of the deep convolutional neural network. For more detail, refer to [1].

To derive the mean and convariance, the element-wise feature map is first given as

$$\mathbf{A}_{j,g}^{l+1}(\mathbf{V}) = b_j^l + \sum_{i=1}^{C^l} \sum_{h=1}^{H^l D^l} \mathbf{W}_{j,i,g,h}^l \phi(\mathbf{A}_{i,h}^l(\mathbf{V})), \tag{12}$$

where l and $l+1$ are indexes of the hidden layers, i and j are indexes of the input and output channels, and h and g denote the location of the element within the input and output channels. When the input image is $\mathbf{V}^{'}$, the element-wise feature map becomes $\mathbf{A}_{j,g}^{l+1}(\mathbf{V}^{'})$.

With equation (12), now we formulate the mean and covariance as in equation (13) and (14), respectively.

$$\mathbb{E}[\mathbf{A}_{j,g}^{l+1}(\mathbf{V})] = \mathbb{E}[\mathbf{b}_j^l] + \sum_{i=1}^{C^l} \sum_{h=1}^{H^l D^l} \mathbb{E}[\mathbf{W}_{j,i,g,h}^l \phi(\mathbf{A}_{i,h}^l(\mathbf{V}))] = 0 \tag{13}$$

$$\begin{aligned} &cov\Big[\mathbf{W}_{j,i,g,h}^l \phi(\mathbf{A}_{i,h}^l(\mathbf{V})), \mathbf{W}_{j,i',g,h'}^l \phi(\mathbf{A}_{i',h'}^l(\mathbf{V}^{'}))\Big] \\ &= \sigma_b^2 + \sigma_w^2 \sum_{h \in g\text{th patch}} \mathbb{E}\big[\phi(\mathbf{A}_{i,h}^l(\mathbf{V}))\phi(\mathbf{A}_{i,h}^l(\mathbf{V}^{'}))\big], \end{aligned} \tag{14}$$

where $\mathbb{E}[.]$ represents the mathematical expectation and $cov[.]$ represents the elemental-wise covariance function. The Gaussian distribution on the weights and feature maps is assumed to be with a zero mean. The convariance includes a term depending on $\phi(\cdot)$. According to [1], a closed form of the covariace can be obtained if $\phi(\cdot)$ is Gaussian and ReLU etc. Please refer to [1] for a solution with ReLU activation.

With equation and (14), we can now consider a deep convolutional neural network as a Gaussian Process. A Deep convolutional neural network is well known for spatial feature extraction, while Gaussian Process is mostly famous for temporal data regression. By regarding a deep convolutional neural network as a Gaussian Process, we can say that in this way, the advantages of the two are combined. Therefore, we apply the method in traffic speed prediction, which essentially needs to take both the spatial and temporal information into consideration in order to achieve high accuracy results. In the next section, we apply two different tasks to evaluate the performances and uncertainties of the approaches introduced in Section III.

III. VALIDATION WITH REAL DATA

A. Data Pre-processes and Preparation

The traffic speed data we use is collected on road segments in the city centre of Santander with 15-minute timesteps for the year 2016. The traffic dataset is provided thanks to the SETA EU project. The traffic speed data structued in equation 1 are devided as below to prepare the training, validating and testing set. Assume that each sample composes of S rows, and 10 samples constitute one loop of data partition.

- Training Set: The first 7 samples (1-7)
- Validating Set: The next 2 samples (8-9)
- Testing Set: The next 2 samples (10)

In this way, unique features of traffic data, such as those during Christmas vacation can be captured.

The CNN, CNN with GP Regression and ConvNet are described in Section II. Subsection B and C are performed to accomplish the following four tasks:

- Task 1: 1-step ahead prediction, with 10-step traffic speed history on 50 road segments.
- Task 2: Based on Task 1, different level of simulated sensor noises are added into the data, and therefore the noisy data becomes $\mathbf{X}_{noisy} = \mathbf{X} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\mu}_{noise}, \boldsymbol{\sigma}_{noise}^2)$.

The models are implemented on Python by using Tensorflow, Keras and GPflow. The training and performance evaluation is running on a PC with 8-core i7-10700K CPU, 48 GB memory and an RTX-2080 GPU. The CNN took 11 ± 0.6 seconds for training and evaluating once, the CNN with GP Regression took 50 ± 9 seconds for training and evaluating once, and the ConvNet took 1.3 ± 0.2 for seconds training and evaluating once.

B. Evaluation and Analyses

In the application of the CNN to traffic prediction problems, we employs the mean squared error (MSE) as the loss function. The Adam optimiser [21] with the exponentially decaying learning rate is utilised to minimise the total MSE. In the application of the CNN with GP Regression, the GP model employs a squared exponential kernel function. For implementation of CNN as shallow GP, the ConvNet GP is the kernel equivalent to a 6-layer CNN that have three convolutional layers, and three max pooling layers. The parameter and layer settings are the same as the typical CNN whose architecture and layer parameters are listed in Table I.

Table I
LAYER PARAMETERS OF CNN

Layer	Parameter	Activation
Convolution1	(256,3,3)	ReLu
Polling1	(2,2)	-
Convolution2	(128,3,3)	ReLu
Polling2	(2,2)	-
Convolution3	(64,3,3)	Relu
Polling3	(2,2)	-
Flatterning	-	-
Fulling-connected	-	-

We employ root mean squared error (RMSE) as the measurement for accuracy of the traffic speed prediction performance. The RMSE is defended as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \widehat{y}_i)^2}{N}}, \tag{15}$$

where y_i is the ground truth speed value and $\widehat{y}_i$ represents the traffic speed prediction on i-th road segment. N denotes the number of the number of the traffic speed data in evaluation set.

Table II
RESULTS VALIDATION (UNIT: KM/H)

	CNN RMSE	CNN with GP Regression RMSE	ConvNet GP RMSE
Task 1	10.04	9.55	8.21

The performances of the networks operating Task 1 is listed in Table II. As shown in Table II, CNN, CNN with GP

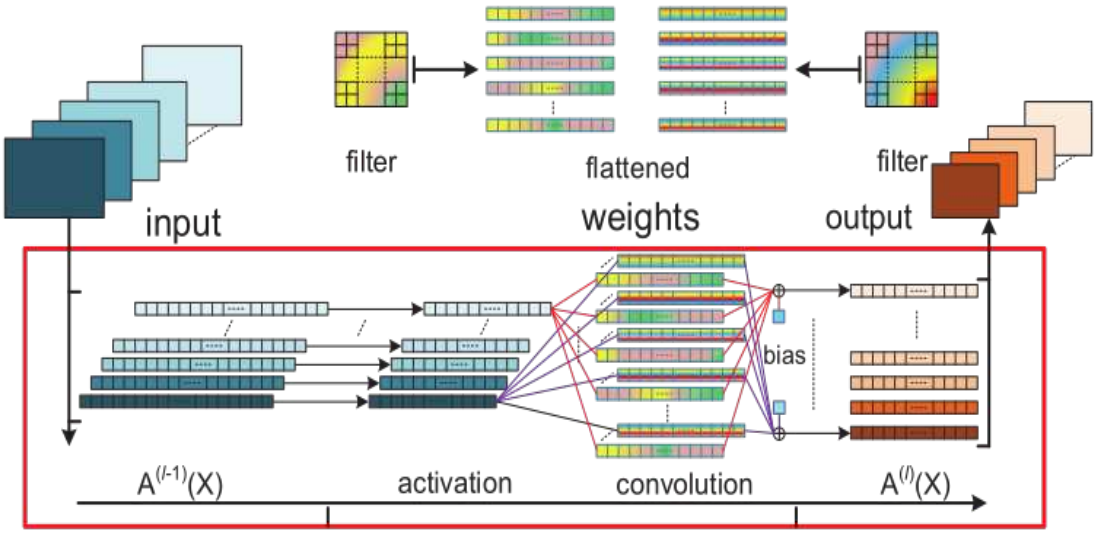

Figure 3. Convolutional Neural Network as Shallow Gaussian Process[1]

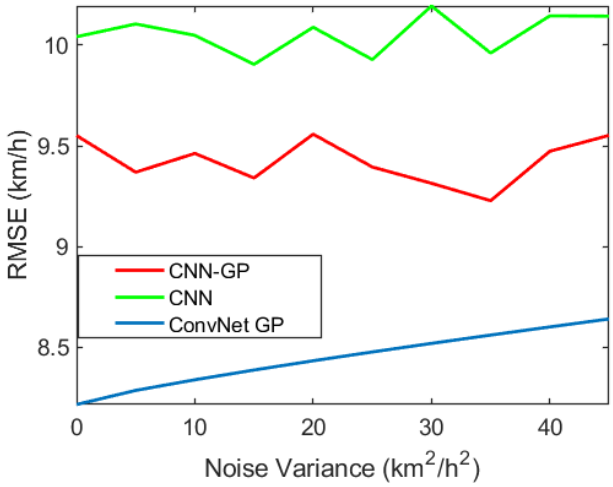

Figure 4. Validation on noisy data. Blue represents the RMSE value obtained from the ConvNet GP; Red represents the RMSE obtained from the CNN-GP framework; and green represents the RMSE obtained from the CNN.

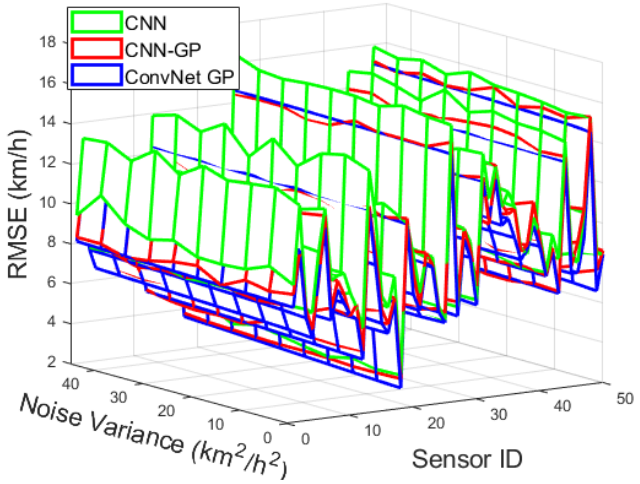

Figure 5. Validation on noisy data. Blue represents the RMSE value obtained from the ConvNet GP; Red represents the RMSE obtained from the CNN-GP framework; and green represents the RMSE obtained from the CNN.

Regression and ConvNet GP are the results trained with our data division algorithm. The GP involved methods, CNN with GP Regression, ConvNet GP, provides smaller RMSE values in Task 1. ConvNet GP provides the most significant difference, 18.23% smaller. Fig. 4 shows the overall performances of the networks operating with different noise levels where the noise variance is from 0 to 45. Generally, CNN with GP Regression and ConvNet GP provides smaller RMSE values than the CNN provides. For ConvNet GP, the RMSE values increase when the noise variance increases. Fig. 5 shows the performances on each sensor with different noise levels are consistent with the overall performance. ConvNet GP provides the smallest RMSE values over all sensors, and the RMSE values increase as the noise variance increases. The performances of the networks keep the same tendency as the sensor ID varies. For example, The largest RMSE values are located on the sensor 26 for all the networks, and the smallest RMSE values are located on the sensor 18 for all the networks.

C. Discussion

Fig. 6 shows the confidence interval of CNN with GP regression with 1σ in red and the confidence interval of GP and Convnet GP with 3σ in purple. The blue dots represent the ground truth. Therefore, the ConvNet GP provides a very narrow confidence interval, and the CNN with GP Regression on the contrary provides a wide confidence interval. To conclude, the ConvNet GP provides the best accuracy with respect to the lowest RMSE but provides the narrowest confidence interval.

IV. CONCLUSIONS AND FUTURE WORK

This work shows that the properties of Gaussian process regression can be beneficial in assessing the impact of sensor data uncertainties in traffic speed prediction. The paper

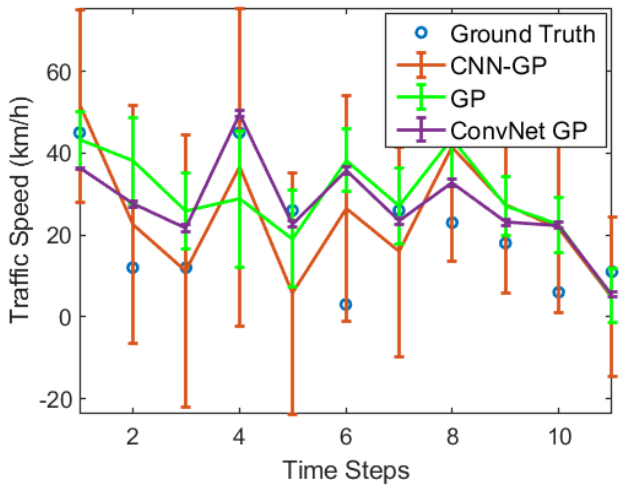

Figure 6. Prediction uncertainty obtained by ConvNet GP. The 1σ confidence interval is applied on CNN with GP regression, 1σ confidence interval is applied on GP and 3σ confidence interval is applied for ConvNet GP.

presents two GP based frameworks, a CNN-GP framework and ConvNet GP. The CNN-GP framework is a CNN with a GP added to quantify the uncertainty of the classification process. The ConvNet GP reformulates a CNN as a Gaussian process. The CNN-GP and ConvNet GP are compared with a baseline generic CNN. The real traffic speed data collected in Santander city, Spain, is used to validate and evaluate the performance of the proposed GP frameworks. The results imply that the ConvNet GP has better capabilities on learning in the presence of uncertainties in the test data and gives 18.23% improvement in the speed RMSE with respect to the generic CNN. With the help of variances provided from the GP, the calculated confidence intervals characterise the credibility of the speed predictions. In this work, we evaluate the performance with different levels of noise in data. In particular, we plan to test more difficult problems, such as traffic speed prediction with the different number of absent sensors. On the other hand, we plan to calculate more proper information bounds for this work, for example, Cramér-Rao bound.

V. ACKNOWLEDGEMENT

We appreciate the support of the SETA project funded by the European Union Horizon 2020 research and innovation program under grant agreement No. 688082. This work was also supported by the UK EPSRC via Grant No. EP/ T013265/1, NSF-EPSRC: "ShiRAS. Towards Safe and Reliable Autonomy in Sensor Driven " jointly with the USA National Science Foundation, Grant No. NSF ECCS 1903466,

REFERENCES

[1] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison, "Deep convolutional networks as shallow Gaussian processes," in *Proc. of the International Conference on Learning Representations*, 2019.

[2] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura, "Traffic state estimation on highway: A comprehensive survey," *Annual Reviews in Control*, vol. 43, pp. 128–151, 2017.

[3] M. S. Ahmed and A. R. Cook, *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*. Transportation Research Board, 1979, no. 722.

[4] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[5] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1. AAAI Press, 2017, pp. 1655–1661.

[6] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[7] Y. Kim, P. Wang, Y. Zhu, and L. Mihaylova, "A capsule network for traffic speed prediction in complex road networks," in *Proc. of the Sensor Data Fusion: Trends, Solutions, Applications (SDF)*. IEEE, 2018, pp. 1–6.

[8] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. of International Conference on Learning Representations*, 2018.

[9] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *ArXiv*, pp. 3856–3866, 2017.

[10] Y. Kim, P. Wang, and L. Mihaylova, "Structural recurrent neural network for traffic speed prediction," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5207–5211.

[11] V. Kumar, V. Singh, P. Srijith, and A. Damianou, "Deep Gaussian processes with convolutional kernels," *ArXiv*, vol. abs/1806.01655, 2018.

[12] P. Wang, N. C. Bouaynaya, L. Mihaylova, J. Wang, Q. Zhang, and R. He, "Bayesian neural networks uncertainty quantification with cubature rules," in *Proc. of the International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[13] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012.

[14] Y. Xie, K. Zhao, Y. Sun, and D. Chen, "Gaussian processes for short-term traffic volume forecasting," *Transportation Research Record*, vol. 2165, no. 1, pp. 69–78, 2010.

[15] P. Wang, Y. Kim, L. Vaci, H. Yang, and L. Mihaylova, "Short-term traffic prediction with vicinity Gaussian process in the presence of missing data," in *Proc. of 2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2018, pp. 1–6.

[16] S. Borgeaud, "Gaussian process classifier for CNN uncertainty," 2018. [Online]. Available: https://github.com/seb5666/cnn_gaussian_process_uncertainty/blob/master/report/report.pdf

[17] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[18] C. E. Rasmussen and C. K. Williams, *Gaussian Process for Machine Learning*. The MIT Press, 2006.

[19] J. Lee, Y. Bahri, R. Novak, S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," *ArXiv*, vol. abs/1711.00165, 2018.

[20] A. Borovykh, "A Gaussian process perspective on convolutional neural networks," *CoRR*, vol. abs/1810.10798, 2018. [Online]. Available: http://arxiv.org/abs/1810.10798

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of the 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980