This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3005429, IEEE Control Systems Letters

Epistemic Uncertainty Quantification in State-space LPV Model Identification Using Bayesian Neural Networks

Yajie Bao, Javad Mohammadpour Velni, and Mahdi Shahbakhti

***Abstract*—This paper presents a variational Bayesian inference Neural Network (BNN) approach to quantify uncertainties in matrix function estimation for the state-space linear parameter-varying (LPV) model identification problem using only inputs/outputs data. The proposed method simultaneously estimates states and posteriors of matrix functions given data. In particular, states are estimated by reaching a consensus between an estimator based on past system trajectory and an estimator by recurrent equations of states; posteriors are approximated by minimizing the Kullback–Leibler (KL) divergence between the parameterized posterior distribution and the true posterior of the LPV model parameters. Furthermore, techniques such as transfer learning are explored in this work to reduce computational cost and prevent convergence failure of Bayesian inference. The proposed data-driven method is validated using experimental data for identification of a control-oriented reactivity controlled compression ignition (RCCI) engine model.**

***Index Terms*—State-space linear parameter-varying model identification, uncertainty quantification, Bayesian neural networks.**

I. INTRODUCTION

DATA-DRIVEN methods have been increasingly developed for global identification of linear parameter-varying (LPV) state-space (SS) models[1]. For LPV-SS identification given only inputs/outputs data, the majority of the current LPV identification methods, including direct prediction-error minimization (PEM) methods and global subspace and realization-based techniques (SID), assume affine scheduling dependency with known basis functions, which restricts the complexity of a representation. The authors in [2] used kernelized canonical correlation analysis (KCCA) to estimate state sequence and then a least-squares support vector machine (LS-SVM) to capture the dependency structure, which suffers from the kernel selection and computational complexity. Moreover, SID techniques first identify a specific IO structure and then construct SS models by either a direct realization or a projection to estimate state sequence, which suffers heavily from the curse of dimensionality, and finally estimate system matrices [3]. The expectation-maximization algorithms estimate states and matrices alternatively [4]. In a very recent work, the authors in [5] have used artificial neural networks to simultaneously estimate states and explore LPV model structural dependency. Most of the available methods in the literature, however, focus on estimating a set of parameters rather than characterizing the statistical properties of the estimation. This approach typically produces good models in the sense of minimizing the expected loss. However, the accuracy under a few operating points can be poor, which can later result in a low-performing controller. Furthermore, robust control techniques cannot be employed without quantifying the uncertainty of estimated model.

In model identification, there are two categories of uncertainty [6]: epistemic uncertainty and aleatoric uncertainty. Epistemic uncertainty is systematic and represented by the uncertainty in model parameters while aleatoric uncertainty is stochastic and representative of the unknowns that result in different system outputs given identical inputs. Multiplicative disturbances in [7] correspond to epistemic uncertainty while additive disturbances correspond to aleatoric uncertainty. The objective of uncertainty quantification is to reduce the epistemic uncertainty to aleatoric uncertainty, as epistemic uncertainty can be reduced by increasing the number of data points while aleatoric uncertainty can be quantified but not reduced [8]. This paper assumes that aleatoric uncertainty is known and aims to quantify epistemic uncertainty given an input/output dataset, as matrix function uncertainties have a significant impact on control design. Additionally, extending the model in this paper to simultaneously capture and quantify these two categories of uncertainties is briefly discussed.

One approach to uncertainty quantification is Bayesian framework. Given a prior distribution of model parameters, the posterior distribution conditioned on a dataset is estimated by maximizing the likelihood of the dataset. The authors in [9] used a recurrent network in a Bayesian framework to perform nonlinear system identification. The authors in [10] used a Gaussian process as a prior distribution to obtain a posterior distribution of coefficient functions given the measured data for identifying LPV models in an input/output (IO) form with an autoregressive with exogenous variable noise structure. The authors in [11] assumed parameter-varying matrices to be component-wise, zero-mean normally distributed and approximated the posterior distribution of system parameters and latent variables via variational inference. However, the linear parameterization of matrix functions restricts model expressiveness of complex systems. Moreover, Gaussian priors can be improper and have negative effects on inference. Additionally, authors in [12] designed deep variational Bayesian filters to improve information content of the latent state-space

This work was financially supported by the United States National Science Foundation under awards #1762595 and #1912757.

Y. Bao and J. Mohammadpour Velni are with the School of Electrical & Computer Engineering, University of Georgia, Athens, GA 30602 yajie.bao@uga.edu, javadm@uga.edu.

M. Shahbakhti is with the Department of Mechanical Engineering, University of Alberta, Edmonton, AB, T6G 1H9 mahdi@ualberta.ca.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3005429, IEEE Control Systems Letters

embedding for state estimation. However, the approach in [12] was not adapted to the general LPV-SS model identification problem.

Inspired by weight uncertainty in neural networks [13] and based on our previous work on identification of state-space LPV models using artificial neural networks (ANN) [5], we propose to use variational Bayesian inference neural networks (BNN), a combination of ANN and variational inference to identify LPV-SS models. Different from [10], [11], [12], the proposed method can simultaneously estimate states, explore arbitrary structural dependency of matrix functions of a representative LPV model and approximate posteriors of the parameters with non-Gaussian priors. States and matrix functions are estimated using State Integrated Matrix Function Estimation (SIME), which is an integrated ANN model in [5] while uncertainties are introduced by assigning each weight of ANN a prior. To approximate posteriors, the main idea is to minimize the Kullback–Leibler (KL) divergence between the parameterized posterior distribution and the true posterior of the LPV model parameters. Monte Carlo sampling is employed to avoid calculating expectation analytically. Sampling also enables non-Gaussian priors. Moreover, reparameterization trick [14] is used for backpropagation to work.

The main contribution of this paper lies in tailoring BNN to the uncertainty quantification of the ANN-based LPV-SS model identification. Remainder of the paper is organized as follows: Section II introduces the problem formulation and variational inference. Combining variational reference and ANN in the context of LPV-SS model identification, i.e., BNN is explained in Section III. Section IV presents model identification results using experimental data. Concluding remarks are finally provided in Section V.

II. Problem Formulation

We consider a discrete-time, state-space LPV model with innovation-type noise as follows:

$$x_{k+1} = A(p_k)x_k + B(p_k)u_k + K(p_k)e_k, \tag{1}$$
$$y_k = C(p_k)x_k + D(p_k)u_k + e_k, \tag{2}$$

where $p_k \in \mathbb{P} \subset \mathbb{R}^{n_p}$, $u_k \in \mathbb{R}^{n_u}$, $x_k \in \mathbb{R}^{n_x}$, $e_k \in \mathbb{R}^{n_y}$, and $y_k \in \mathbb{R}^{n_y}$ denote the scheduling variables, inputs, states, stochastic white noise process, and outputs of the system at time instant k, respectively, and A, B, C, D, and K are smooth matrix functions of p_k. In [5], each of the functions was represented by a fully-connected ANN. By substituting e_k from (2) into (1), we obtain

$$x_{k+1} = \tilde{A}(p_k)x_k + \tilde{B}(p_k)u_k + K(p_k)y_k, \tag{3}$$
$$y_k = C(p_k)x_k + D(p_k)u_k + e_k, \tag{4}$$

where $A(p_k) = \tilde{A}(p_k) + K(p_k)C(p_k)$ and $B(p_k) = \tilde{B}(p_k) + K(p_k)D(p_k)$. Instead of estimating a deterministic set of matrix functions as in [5], all weights in the ANNs are mutually independent random variables such that matrix functions are also random variables. BNN is employed to estimate posteriors for the weights. The posteriors of the matrix functions can then be estimated using Monte Carlo method. Therefore, the problem of LPV-SS model identification with uncertainty quantification is to estimate states (if they are unknown) and posteriors of $\tilde{A}(p_k)$, $\tilde{B}(p_k)$, $C(p_k)$, $D(p_k)$ and $K(p_k)$ given the measurements $\mathcal{D} = \{u_k, y_k, p_k\}_{k=1}^{N_\mathcal{D}}$[1].

A. Variational Inference

Variational inference is a machine learning technique to approximate difficult-to-compute probability density functions by finding a member from a family of densities that is closest to the target in the sense of KL divergence [15]. Using Bayes' formula, we have

$$p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} \propto p(\mathcal{D}|w)p(w),$$

where w denotes all the parameters of a model. However, $p(w|\mathcal{D})$ is hard to calculate analytically especially when the probability graph is complex such as an ANN with the weights being random variables. To circumvent this, variational inference computes

$$\theta^* = \arg\min_\theta \mathrm{KL}\Big(q(w;\theta)\|p(w|\mathcal{D})\Big) \tag{5}$$
$$= \arg\min_\theta \mathrm{KL}\Big(q(w;\theta)\|p(w)\Big) - \mathbb{E}_{q(w;\theta)}\left[\log p(\mathcal{D}|w)\right]$$
$$= \arg\min_\theta \Big(\mathbb{E}_{q(w;\theta)}\left[\log q(w;\theta)\right] - \mathbb{E}_{q(w;\theta)}\left[\log p(w)\right] - \mathbb{E}_{q(w;\theta)}\left[\log p(\mathcal{D}|w)\right]\Big), \tag{6}$$

where $q(w;\theta)$ is a family of distributions parameterized by θ, $p(w|\mathcal{D})$ is the true posterior, and $p(w)$ is the prior. The cost function (5), known as the evidence lower bound (ELBO) [15], provides a trade-off between believing priors and fitting data (the second term).

B. LPV-SS Model Identification Using ANN

In [5], we proposed State Integrated Matrix Function Estimation (SIME), an integrated ANN model, to estimate states and explore structural dependency of matrix functions simultaneously. SIME solves the following problem:

$$\min \Big(\gamma_1 \sum_{k=d+1}^{N-1} \|\hat{x}_{k+1}^{(1)} - \hat{x}_{k+1}^{(2)}\|_2^2 + \gamma_2 \sum_{k=d+1}^{N-1} \|\hat{y}_k - y_k\|_2^2 + \gamma_3 \sum_{k=d+1}^{N-1} \|\hat{y}_{k+1} - y_{k+1}\|_2^2\Big) \tag{7}$$
$$\text{s.t. } \hat{x}_{k+1}^{(1)} = f_{D_{\tilde{A}}}(p_k)\hat{x}_k^{(2)} + f_{D_{\tilde{B}}}(p_k)u_k + f_{D_K}(p_k)y_k, \tag{8}$$
$$\hat{x}_k^{(2)} = f_{D_x}(\overline{z}_k^d), \quad \hat{x}_{k+1}^{(2)} = f_{D_x}(\overline{z}_{k+1}^d), \tag{9}$$
$$\hat{y}_k = f_{D_C}(p_k)\hat{x}_k^{(2)}, \quad \hat{y}_{k+1} = f_{D_C}(p_{k+1})\hat{x}_{k+1}^{(1)}. \tag{10}$$

$\overline{z}_k^d := \begin{bmatrix}\overline{p}_k^{d\mathrm{T}} & \overline{u}_k^{d\mathrm{T}} & \overline{y}_k^{d\mathrm{T}} & p_k^{\mathrm{T}} & u_k^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$ represents the past and current information at time instant k, where $\overline{p}_k^d := \begin{bmatrix}p_{k-d}^{\mathrm{T}} & \cdots & p_{k-1}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$ denotes the past scheduling trajectory, and $\overline{u}_k^d$ and $\overline{y}_k^d$ are defined similarly. Furthermore, $x^{(2)}$ is an estimator inspired by

$$x_k = \underbrace{\left(\prod_{i=1}^{d} \tilde{A}(p_{k-i})\right) x_{k-d}}_{\mathcal{X}_p^d(k)} + \mathcal{R}_p^d(k)\overline{u}_k^d + \mathcal{V}_p^d(k)\overline{y}_k^d,$$

[1]We use identical notations for random variables and samples, which can be inferred from the context.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3005429, IEEE Control Systems Letters

as derived in [2] while $x^{(1)}$ is by the recurrent equation of states. Based on Lemma 3.1 in [2], $d \geq n_x$. A larger value of d indicates using more information but also increases computational cost. In experiments, d is determined using cross validation. Finally, $f_{D_i}, i = \tilde{A}, \tilde{B}, K, C, D, x$ are represented by fully-connected ANNs to approximate true matrix functions and states.

The outcome of SIME is a point estimate of matrix functions, i.e., a set of w, from the perspective of parameter estimation. Minimizing mean squared error (MSE) is equivalent to maximizing the likelihood of data by viewing $p(y|p, z, u; w)$ as a Gaussian distribution. Moreover, maximum likelihood estimation can be seen as a special case of the maximum a posteriori (MAP) estimation that assumes a uniform prior distribution of the parameters. Furthermore, by using ℓ_2 regularization with respect to w, we can obtain a MAP estimation of w with a Gaussian prior, which can be seen from

$$\begin{aligned}\max_w \log p(w|\mathcal{D}) &= \max_w \Big(\log p(\mathcal{D}|w) + \log p(w) \Big) \\ &= \max_w \Big(\log p(\mathcal{D}|w) + \|w\|_2^2 \Big).\end{aligned}$$

This paper aims to estimate $p(w|\mathcal{D})$ instead of one set of w^* that minimizes the expected loss.

III. Bayesian Neural Network (BNN)

In this section, we first introduce the necessary tools that enable ANNs to do variational inference by backpropagation, which results in a DenseVariational layer [13], a key component of BNN. Then, we discuss how to combine DenseVariational layer with SIME for LPV-SS model identification.

A. Variational Inference in ANNs

First, we consider the forward-pass and backward-pass in an ANN training iteration separately. For the forward pass, cost function (6) is evaluated. As $q(w; \theta)$ cannot be expressed in closed form and integral operation is prohibitive in ANN, we estimate (6) by

$$\frac{1}{N} \sum_{i=1}^{N} \Big[\log q(w^{(i)}; \theta) - \log p(w^{(i)}) - \log p(\mathcal{D}|w^{(i)}) \Big] \quad (11)$$

where $\{w^{(i)}\}_{i=1}^N$ are i.i.d. samples drawn from the parameterized posterior $q(w; \theta)$. For the backward pass, gradients of (6) are required for optimization. However, when computing the gradient of the first term in (6)

$$\begin{aligned}\bigtriangledown_\theta \mathbb{E}_{q(w;\theta)} \left[\log q(w; \theta) \right] = &\int_w \bigtriangledown_\theta \log q(w; \theta) \bigtriangledown_\theta q(w; \theta) \mathrm{d}w \\ &+ \mathbb{E}_{q(w;\theta)} \left[\bigtriangledown_\theta \log q(w; \theta) \right], \quad (12)\end{aligned}$$

the first term in (12) is not an expectation, which impedes the estimation of the gradients using Monte Carlo methods and hence the backpropagation of a training iteration. To tackle this problem, a reparameterization trick is employed. In particular, a sample ϵ is drawn from a known distribution and then transformed by a deterministic function $t(\theta, \epsilon)$ for which a gradient can be computed. The known distribution and transform function determine $q(w; \theta)$. For example, if $\epsilon \sim \mathcal{N}(0, I)$ and $t = \mu + \sigma \bigodot \epsilon$ where $\bigodot$ denotes element-wise multiplication, then $q(w; \mu, \sigma)$ is Gaussian. This reparameterization trick provides an unbiased gradient estimation of (6) with respect to θ, as shown by *Proposition 1* in [13]. Additionally, the variance of y_k can be decomposed as

$$\begin{aligned}\mathrm{var}(y_k) \approx \sigma^2 &+ \frac{1}{N} \sum_{i=1}^{N} \hat{y}_k(\mathrm{x}; w^{(i)})^{\mathrm{T}} \hat{y}_k(\mathrm{x}; w^{(i)}) \\ &- \left(\frac{1}{N} \sum_{i=1}^{N} \hat{y}_k(\mathrm{x}; w^{(i)}) \right)^{\mathrm{T}} \left(\frac{1}{N} \sum_{i=1}^{N} \hat{y}_k(\mathrm{x}; w^{(i)}) \right), \quad (13)\end{aligned}$$

where x denotes all the inputs to the neural networks and

$$\begin{aligned}\hat{y}_k(\mathrm{x}; w^i) = &\frac{1}{N} \sum_{i=1}^{N} f_{D_C}(p_k) \times \\ &\Big(f_{DV_{\tilde{A}}}(p_{k-1}; w^{(i)}) \cdot f_{D_x}(z_{k-1}) + f_{D_B}(p_{k-1}) u_{k-1} \Big)\end{aligned}$$

in the context of LPV-SS model identification using BNN and SIME[2]. The third term in (11), named the negative log-likelihood loss, can be used to quantify aleatoric uncertainty σ in (13). For example, when aleatoric uncertainty is assumed to be white noise, then $p(\mathcal{D}|w^{(i)}) \sim \mathcal{N}(\mu_a, \sigma_a^2)$, where μ_a and σ_a^2 can be learned from data. Moreover, heteroscedastic aleatoric uncertainty can be expressed using a parameterized function of inputs to represent σ_a [8]. However, simultaneously quantifying epistemic uncertainty and aleatoric uncertainty can increase the difficulty of inference and cause over-fitting problem. Decomposition of uncertainties is required [16], which is beyond the scope of this paper and will be examined in a future work.

Another advantage of using Monte Carlo method is that we can use arbitrary priors with/without tractable marginal distributions. In Bayesian methods, priors encode the information known before any evidence is presented and provide a means for stabilizing inferences in complex, high-dimensional problems [17]. More importantly, priors can affect the parameterized posterior and thus the predictive posterior [18] by the KL divergence loss (6). On prior selection, there are many schools of thoughts, such as informative, weakly informative and noninformative priors. Providing a prior for each weight to constitute a meaningful prior of the represented function by ANNs is intractable. Using a trainable prior is possible but criticized for employing data twice. Instead, the scale mixture of two Gaussian densities

$$p(w) = \prod_j \pi \mathcal{N}(w_j|0, \sigma_1^2) + (1 - \pi) \mathcal{N}(w_j|0, \sigma_2^2), \quad (14)$$

was proposed in [13] to avoid the need for prior parameter optimization based upon training data. This prior can represent both a heavy tail by a large σ_1 and concentration by a small σ_2.

Combining the components described above, a DenseVariational layer is composed of priors $p(w_j)$, parameterized posteriors $p(w_j; \theta)$ and estimated $\mathbb{E}_{q(w_j;\theta)} \left[\log q(w_j; \theta) - \log p(w_j) \right]$ which are added to the cost function of the ANNs (see Figure 1). Here, we use

[2]We use subscript DV to distinguish BNNs composed of DenseVariational layers from ANNs composed of Dense layers.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3005429, IEEE Control Systems Letters

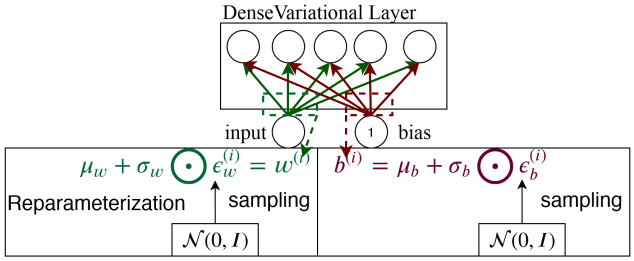


Fig. 1: Schematic diagram of DenseVariational layer, where μ_w, σ_w, μ_b and σ_b are trainable parameters while Dense layers only have w and b. Note that $w^{(i)}$ and $b^{(i)}$ can be seen as samples from posterior distributions. All parameters are initialized according to priors.

$t = \mu + \sigma \odot \epsilon$ as the transform function to parameterize posteriors and attribute the expressiveness of the parameterized posterior of $f_{DV_{\tilde{A}}}$ to the compositions of w_j. Compared with a Dense layer (i.e., a fully-connected layer), two more sets of parameters (μ, σ) are required along with weights w and biases b if we use fixed priors. For implementation of the DenseVariational layer, interested reader is referred to [19], in which the authors provide Bayesian layers, a module for neural network uncertainty, to expedite the experimentation with neural network uncertainty.

To predict outputs using the trained BNN, samples are drawn from the posteriors of weights and used to compute the output value of the network, which is equivalent to predicting outputs using neural networks drawn from the selected hypothetical ensemble. Statistics such as the mean and standard deviation can be computed using the predictions. The number of samples is determined to guarantee a stable estimation. Monte Carlo methods enable Bayesian inference for ANNs but also pose challenges to the broad application of BNNs, such as the high computational cost and convergence failure [20]. In the next subsection, we will tackle these challenges in the context of LPV-SS model identification problem.

B. BNN in LPV-SS Model Identification

In this section, we discuss techniques that facilitate the training of BNNs for quantifying uncertainty in LPV-SS model identification.

This work aims to quantify uncertainties specifically in matrix functions. If a single DenseVariational layer with a linear activation function is used to represent a matrix function (e.g., $f_{DV_{\tilde{A}}}$), then, the model degrades to be a Gaussian Process and $f_{DV_{\tilde{A}}}|\mathcal{D}$ follows Gaussian distribution as $w_j|\mathcal{D}$ are independently distributed and follow Gaussian distribution. However, if we use multiple DenseVariational layers with non-linear activation functions, the explicit analytical form of $q(f_{DV_{\tilde{A}}}|\mathcal{D})$ is intractable while Monte Carlo sampling can be used to estimate the variance of $f_{DV_{\tilde{A}}}|\mathcal{D}$ and thus the variance of $\hat{y}$.

1) Trade-off between bias and variance: As variational inference is done layer-wise, it is straightforward to replace all f_{D_i} with f_{DV_i}. However, such replacement significantly increases model complexity and thus the probability of arriving at bad local minima. Instead, a more reasonable approach is to only replace $f_{D_{\tilde{A}}}$ with $f_{DV_{\tilde{A}}}$. From the perspective of LPV modeling, uncertainty in $\tilde{A}(p_k)$ has a larger impact on the system description than other matrix functions especially when states x_k are unknown and estimated from inputs/outputs data. Moreover, SIME in [5] uses the consensus of two estimators of one state to facilitate the state and matrix function estimation. Using DenseVariational layers for both estimators can cause instability of training and even failure of convergence.

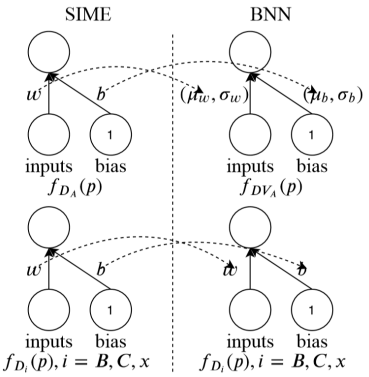


Fig. 2: Schematic diagram of the proposed transfer learning approach.

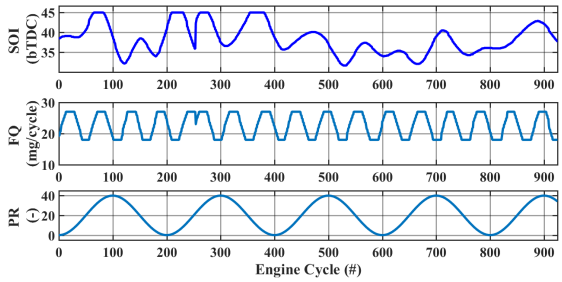
2) Transfer learning by fine-tuning: Generally, Bayesian models cannot achieve an accuracy as high as that of a deterministic model given a dataset. A question arises here that how to quantify uncertainty of the deterministic model. As SIME and BNN can share the same ANN structures, one can transfer parameters (w^*_{SIME}, b^*_{SIME}) of a trained SIME to the corresponding BNN, fix those parameters in $f_{DV_{\tilde{A}}}$ and train the parameters related to variance evaluation (i.e., σ_w and σ_b) and other parameters (see Figure 2). In particular, when initializing the BNN, we replace the randomly initialized μ_{w_0} and μ_{b_0} by (w^*_{SIME}, b^*_{SIME}). Moreover, when training the BNN, we can choose to fix or fine-tune μ_w and μ_b while σ_w and σ_b are always trainable for capturing uncertainty. This transfer learning technique, named fine-tuning, can accelerate the training process by providing a good initialization and achieve better accuracy compared to training from the scratch. In addition, all the parameters can be fine-tuned after parameter transfer.

IV. Experimental Results and Validation

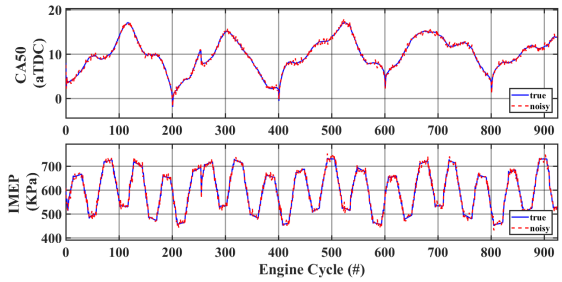
In this section, we use data collected from a high fidelity simulation model of a reactivity controlled compression ignition (RCCI) engine to validate our proposed method.

1) Data generation: The data consists of measurements of the control inputs, scheduling variable, and measurement outputs which are as follows: $U = [SOI \quad FQ]^{\mathrm{T}}$, $p = PR$, $Y = [CA50 \;\; IMEP]^{\mathrm{T}}$, where SOI is start of injection; FQ is fuel quantity; PR is premixed ratio of dual fuel; $CA50$ is the engine combustion phasing; $IMEP$ is indicated mean effective pressure of the engine. The states of the model are assumed to be not available for measurement. Figure 3 shows the signals used to generate an input/output dataset. Additionally, a normally distributed measurement noise is added to the outputs to maintain SNR $= 20$ db. The noisy dataset contains 926 operating points and is split into a training set and a testing set with a splitting ratio of $65\%/35\%$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3005429, IEEE Control Systems Letters



(a) Inputs to the engine.



(b) Outputs from the engine.

Fig. 3: Input/output dataset used for LPV-SS model learning of RCCI engine. The blue line in (b) shows the true outputs and red dashed line shows the noisy outputs.

2) Technical details of BNN: For all the DenseVariational layers we used in the experiments, the hyperparameters in the priors (14) are set as $\pi = 0.5$, $\sigma_1^2 = 1.5$ and $\sigma_2^2 = 0.1$ for simplicity, although each layer can have respective hyperparameter settings. We used a BNN with three DenseVariational layers to represent $f_{DV_{\tilde{A}}}$. For $f_i, i = \tilde{B}, C, K, x$, as discussed in Section III-B1, ANNs with Dense layers are adopted and the number of Dense layers are considered to be 3, 4, 3, and 5, respectively. Moreover, all the layers use Exponential Linear Units (ELU) as activation function except for the last layer of each neural network[3]. Additionally, we implemented the model with all the matrix functions represented by BNN for comparison. $\text{var}(\mathcal{D}|w)$ is assumed to be 1.

For model optimization, we use Adam optimizer in Keras [21]. The learning rate of Adam is set to be 0.0001 and decay to be $1e-6$. All the other parameters of Adam are set as default. We trained the complete model for $3,000$ epochs with batch size of 1. Larger batch size can be used but may slightly lower performance for regression, as the samples in one batch can have different scheduling variables. Using cross-validation, the weights of three loss functions were determined to be 0.1, 1 and 1. Additionally, the experiments are conducted on a computer with a 3.6 GHz CPU and 8 GB RAM.

3) Results and discussion: First, we consider using the following model

$$\begin{aligned} \hat{x}_{k+1}^{(1)} &= f_{DV_A}(p_k)\hat{x}_k^{(2)} + f_{D_B}(p_k)u_k, \\ \hat{y}_k &= f_{D_C}(p_k)\hat{x}_k^{(2)}, \quad \hat{y}_{k+1} = f_{D_C}(p_{k+1})\hat{x}_{k+1}^{(1)}, \end{aligned} \tag{15}$$

to fit data. The idea is to further reduce model complexity by removing f_{D_k} in (8). The results are shown in Figure 4. The

[3]We refer to SIME proposed in [5] to design the model.

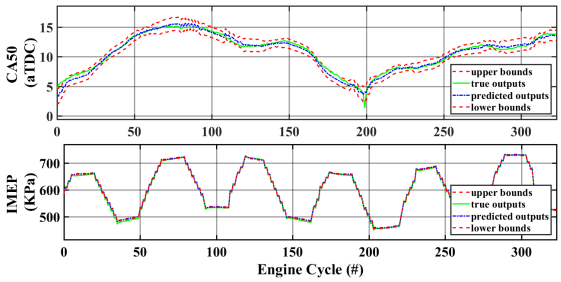


Fig. 4: The area between the two dashed red lines is within 2 estimated standard deviations of estimated mean (the blue line), which is about 95% confidence interval. It is noted that $\text{BFR}_{CA50} = 86.65\%$ and $\text{BFR}_{IMEP} = 95.90\%$ using the estimated mean as predictions for outputs.

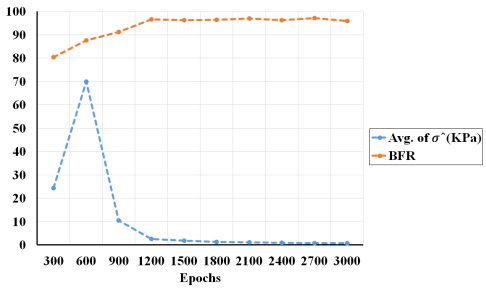


Fig. 5: BFR_{IMEP} and average of $\hat{\sigma}_{IMEP}$ (in KPa).

Best Fit Ratio (BFR) is calculated according to

$$\text{BFR}(\theta) = 100\% \cdot \max_k \left(1 - \frac{\|y_k - \hat{y}_k(\theta)\|_2}{\|y_k - \bar{y}\|_2}, 0\right), \tag{16}$$

where $\bar{y}$ denotes the mean value. Only 3.4% (11 out of 324) of outputs in the testing set are outside $2\hat{\sigma}_{CA50}$ of $\hat{\bar{y}}_{CA50}$, which shows the proposed method has a very good generalization performance. For IMEP, as BFR_{IMEP} is pretty high, $\hat{\sigma}_{IMEP}$ is close to zero and of little significance for control design purposes. Figure 5 shows that the average of estimated standard deviation decreases and the BFR of IMEP increases as the epoch increases, which shows that optimizing the BNN can lead to a better model with higher confidence. Estimating statistics of $f_{DV_{\tilde{A}}}$ is similar to estimating the outputs using sampling, as $f_{DV_{\tilde{A}}}$ is the output of a BNN.

Moreover, we tested a few varieties of Model (15) and the results are summarized in Table I. Model-α replaces $f_{D_i}, i = \tilde{B}, C$ in (15) with $f_{DV_i}, i = B, C$ while Model-β adds f_{D_K} to Model (15). Model-γ further transferred and fine-tuned the parameters from the solution to Problem (7) to Model-β. As Monte Carlo sampling is adopted to estimate mean and variance, BFR and $\hat{\sigma}$ can vary slightly with each run of evaluation process which samples 500 times. For each model, we run evaluation for 10 times and use the average of 10 evaluation results as the final performance measure of the identified model. Comparing the results of Model-α and Model (15) shows that using BNNs to represent all matrix functions resulted in a worse nominal model with larger variance, which confirms our discussion in Section III-B1. Additionally, Model-α costs 14 times more training time than Model (15). Moreover, Model-β considers innovation-type noise but shows

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LCSYS.2020.3005429, IEEE Control Systems Letters

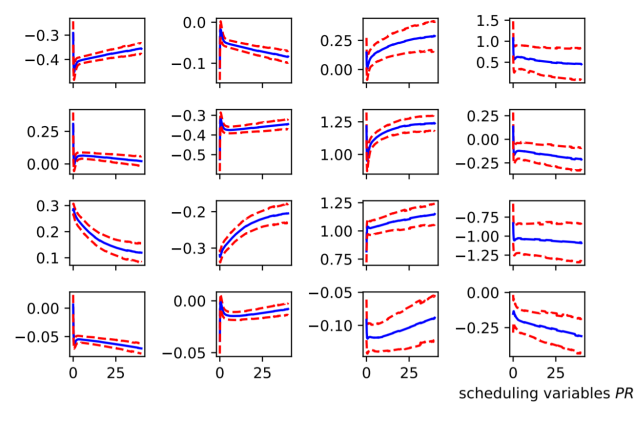


Fig. 6: Estimated mean and 95% confidence interval of $f_{DV_{\tilde{A}}}(p)$ using transfer learning without tuning μ. The (i,j)-th subplot shows function $\tilde{A}[i,j](p)$.

TABLE I: Results of model varieties

	Model (15)	Model-α	Model-β	Model-γ
BFR$_{CA50}$	86.65%	76.27%	79.17%	87.46%
Avg. of $\hat{\sigma}_{CA50}$	0.4011	0.5283	0.5347	0.3735
BFR$_{IMEP}$	95.90%	93.80%	92.07%	97.38%
Avg. of $\hat{\sigma}_{IMEP}$	0.7078	2.6994	0.5236	0.7888

no advantage over Model (15), which indicates that complexity is not necessarily positively correlated with performance. However, by using transfer learning approach discussed in Section III-B2, Model-γ shows a significant enhancement over Model-β and beats Model (15) by a narrow margin. Figure 6 shows the estimated mean and 95% confidence interval of $f_{DV_{\tilde{A}}}(p)$ using transfer learning without tuning μ of all the weights. The estimated mean $\overline{f}_{DV_A}$ is almost identical to the result in [5], which shows that Monte Carlo sampling with 500 samples is sufficient to estimate the mean of f_{DV_A}.

In addition, on the noisy dataset, deterministic SIME model achieves 70.05% BFR for CA50 and 96.32% for IMEP using the reported hyperparameters in [5]. This result, compared with Model (15), shows that BNNs are better at handling noise. The BFR of CA50 can be increased to 83.83% and IMEP to 97.33% by fine-tuning the pre-trained model on the dataset shown in Figure 3 while Model-γ gives similar accuracy with uncertainty quantification.

V. Concluding Remarks

In this paper, a BNN approach was proposed to quantify uncertainties in identified LPV-SS models only using inputs/outputs data. Specifically, a DenseVariational layer was introduced to do the variational inference layer-wise. Each weight of this layer was considered to be a random variable with a scale mixture Gaussian densities as the prior and a parameterized Gaussian density as the posterior. By minimizing the KL divergence between parameterized posteriors and the true posteriors, and the MSE between the predicted outputs and the true outputs, the proposed method was shown to provide a robust nominal model with uncertainty quantification. Experiments on a high-fidelity RCCI engine model validated the effectiveness of the proposed method. In particular, the identified LPV-SS model of the engine could achieve a comparative BFR against deterministic SIME model on the data without noise while it was shown to be more robust to noise at different SNRs.

References

[1] P. B. Cox, "Towards efficient identification of linear parameter-varying state-space models," Ph.D. dissertation, Ph. D. dissertation, Eindhoven University of Technology, 2018.

[2] S. Z. Rizvi, J. Mohammadpour Velni, F. Abbasi, R. Tóth, and N. Meskin, "State-space LPV model identification using kernelized machine learning," *Automatica*, vol. 88, pp. 38–47, 2018.

[3] P. B. Cox, R. Tóth, and M. Petreczky, "Towards efficient maximum likelihood estimation of LPV-SS models," *Automatica*, vol. 97, pp. 392–403, 2018.

[4] A. Wills and B. Ninness, "System identification of linear parameter varying state-space models," in *Linear parameter-varying system identification: new developments and trends*. World Scientific, 2012, pp. 295–315.

[5] Y. Bao, J. Mohammadpour Velni, A. Basina, and M. Shahbakhti, "Identification of state-space linear parameter-varying models using artificial neural networks," in *21st IFAC World Congress* (accepted and to appear). IFAC, 2020.

[6] H. G. Matthies, "Quantifying uncertainty: Modern computational representation of probability and applications," in *Extreme Man-Made and Natural Hazards in Dynamics of Structures*, A. Ibrahimbegovic and I. Kozar, Eds. Springer Netherlands, 2007, pp. 105–135.

[7] Q. Cheng, M. Cannon, B. Kouvaritakis, and M. Evans, "Stochastic MPC for systems with both multiplicative and additive disturbances," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 2291–2296, 2014.

[8] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[9] A. Brusaferri, M. Matteucci, P. Portolani, and S. Spinelli, "Nonlinear system identification using a recurrent network in a Bayesian framework," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 319–324.

[10] A. Golabi, N. Meskin, R. Tóth, and J. Mohammadpour, "A Bayesian approach for LPV model identification and its application to complex processes," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2160–2167, 2017.

[11] C. O. Becker and V. M. Preciado, "Variational inference for linear systems with latent parameter space," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 5662–5667.

[12] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," 2016.

[13] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," 2015.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013.

[15] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, p. 859–877, Feb 2017. [Online]. Available: http://dx.doi.org/10.1080/01621459.2017.1285773

[16] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning," *arXiv preprint arXiv:1710.07283*, 2017.

[17] A. Gelman, D. Simpson, and M. Betancourt, "The prior can often only be understood in the context of the likelihood," *Entropy*, vol. 19, no. 10, p. 555, 2017.

[18] H. K. Lee, "Neural networks and default priors," in *Proceedings of the American Statistical Association*, 2005.

[19] D. Tran, M. W. Dusenberry, M. van der Wilk, and D. Hafner, "Bayesian layers: A module for neural network uncertainty," *CoRR*, vol. abs/1812.03973, 2018. [Online]. Available: http://arxiv.org/abs/1812.03973

[20] T. Papamarkou, J. Hinkle, M. T. Young, and D. Womble, "Challenges in Bayesian inference via Markov chain Monte Carlo for neural networks," 2019.

[21] F. Chollet *et al.*, "Keras," https://keras.io, 2015.