



Statistical Parametric Speech Synthesis Using Deep Gaussian Processes

Tomoki Koriyama , *Member, IEEE*, and Takao Kobayashi , *Fellow, IEEE*

Abstract—This paper proposes a framework of speech synthesis based on deep Gaussian processes (DGPs), which is a deep architecture model composed of stacked Bayesian kernel regressions. In this method, we train a statistical model of transformation from contextual features to speech parameters in a similar manner to deep neural network (DNN)-based speech synthesis. To apply DGPs to a statistical parametric speech synthesis framework, our framework uses an approximation method, doubly stochastic variational inference, which is suitable for an arbitrary amount of data. Since the training of DGPs is based on the marginal likelihood that takes into account not only data fitting, but also model complexity, DGPs are less vulnerable to overfitting compared with DNNs. In experimental evaluations, we investigated a performance comparison of the proposed DGP-based framework with a feedforward DNN-based one. Subjective and objective evaluation results showed that our DGP framework yielded a higher mean opinion score and lower acoustic feature distortions than the conventional framework.

Index Terms—Statistical parametric speech synthesis, Gaussian process, stochastic variational inference, Bayesian model.

I. INTRODUCTION

IN STATISTICAL parametric speech synthesis (SPSS), the relationships between acoustic and linguistic features are expressed with statistical models. Acoustic features are speech parameters such as mel-cepstral coefficients, f_0 values, phone durations, and linguistic features are referred to as contexts. Models with deep architecture have recently become widely used for research on SPSS, especially with the introduction of deep neural networks (DNNs) [1]. After the proposal of DNN-based speech synthesis, various extensions, such as the long short-term memory recurrent neural network (LSTM-RNN)-[2] and attention-based synthesis [3], [4], have been developed. One of the advantages of using DNN is that it automatically extracts an appropriate representation of complex dependencies between acoustic features and contextual factors. Another advantage is that the model can be trained efficiently by minibatch-based optimization even if the amount of training data is huge. However, DNN-based frameworks often encounter an overfitting problem, and the performance largely depends on the choice

of meta-parameters including model topology and training configuration, especially for a limited amount of training data.

Nevertheless, deep models have the potential to generate natural-sounding synthetic speech. In this context, we focus on deep Gaussian process (DGP) [5], which is composed of stacked multiple Gaussian processes (GPs). A single-layer GP regression is regarded as Bayesian kernel regression. Kernel regression can perform nonlinear transformation with an infinite number of basis functions using a small number of hyperparameters, whereas a neural network generally requires a large number of parameters. However, single-layer GPs have limitations caused by their kernel functions. It is assumed that with a GP using a widely used radial basis function (RBF) kernel, for example, the covariance between two points depends on the Euclidean (or Mahalanobis) distance of their inputs. This is not always appropriate for complicated input features such as contexts in SPSS. DGP is expected to overcome the limitation in the expressiveness of the kernel function of a single layer GP by stacking GPs. Moreover, since the fitting process in DGP is performed by maximizing marginal likelihood (a.k.a. empirical Bayes), DGP is less vulnerable to overfitting than DNNs trained by maximum likelihood or maximum a posteriori criterion.

Although the original DGP framework [5] was introduced for small-scale training data, recent studies have proposed DGP frameworks suitable for an arbitrary amount of data [6]–[9]. In this paper, we focus on DGP with doubly stochastic variational inference (DSVI) [9], which has no limits in the choice of kernel functions and is easy to implement. DSVI-DGP is composed of stochastic variational inference (SVI) and Monte Carlo sampling known as the reparametrization trick [10]. SVI is a variational approximation method that enables parameter optimization using stochastic gradient descent algorithms including AdaGrad [11] and Adam [12] which are widely used in deep learning frameworks.

In this paper, we incorporate DSVI-DGP into SPSS. We experimentally examine the effects of meta-parameters and model configurations, such as the number of layers, kernel functions, hidden layer dimensionality, and inducing points used for SVI. We also compare the synthetic speech of the proposed framework with that of a feed-forward DNN framework, which is a foundation of various neural network models. In addition, we compare another GP-based framework [13], [14] that uses a decision tree instead of a deep architecture.

The rest of the paper is organized as follows. We describe a single layer GP and its approximation method of SVI, and DSVI-DGP in Sections III and IV, respectively. In Section V, we

Manuscript received July 20, 2018; revised December 5, 2018 and February 19, 2019; accepted February 26, 2019. Date of publication March 14, 2019; date of current version April 4, 2019. This work was supported in part by JSPS Grants-in-Aid for Scientific Research (KAKENHI) Grant Number 15H02724, 17K12711. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zhen-Hua Ling. (*Corresponding author: Tomoki Koriyama.*)

The authors are with the School of Engineering, Tokyo Institute of Technology, Yokohama 226-8502, Japan (e-mail: t.koriyama@ieee.org; takao.kobayashi@ip.titech.ac.jp).

Digital Object Identifier 10.1109/TASLP.2019.2905167

explain incorporating DSVI-DGP into SPSS. Kernel functions investigated as a covariance function of each DGP layer are discussed in Section VI. In Section VII, we present the results of objective and subjective evaluations of the proposed framework. We discuss future directions in Section VIII and conclude the paper in Section IX.

II. RELATED WORK

A. GP-Based Speech Synthesis With Decision-Tree Context Clustering

The use of GP for SPSS was proposed in [13], [14]. This framework is focused on the exemplar-based nature of GP regression which is derived from the nonparametric model with an infinite number of basis functions. GP regression can use all acoustic feature sequences in training data without parameterizing them into a fixed parameter size. To enable exemplar-based regression, a Gram matrix is approximated by the sum of block-diagonal and low-rank matrices, known as partially independent conditional (PIC) approximation [15]. Although it has been reported that this PIC-GP-based framework yielded comparable results with a DNN-based one for not only Japanese [16] but also Thai [17], the following drawbacks still remain. One is the requirement of data partition for the block-diagonal approximation. To partition data, we used context-dependent decision tree clustering in a similar manner to that used in hidden Markov model (HMM)-based speech synthesis. This implies that the model performance is limited by the decision tree. The decision trees are also constructed separately for respective speech parameters, such as f_o and mel-cepstrum; thus, it is difficult to train a multi-stream model in the PIC-GP-based framework.

The concept of the proposed DGP-based framework is different from the PIC-GP-based one. The proposed framework is focused on a Bayesian framework of GP regression rather than the exemplar-based nature. Instead of clustering, we use a deep architecture, which not only eliminates the limitations of an HMM and decision tree but also enables multi-stream training. Moreover, since deep architectures can extract structural features automatically, we can use the same simple input features as those of DNN-based speech synthesis.

B. Speech Representation and Synthesis by GPs

GPs have been used in many speech representation and synthesis studies. Henter *et al.* [18] proposed a representation framework based on Gaussian process dynamical model (GPDM), which encodes acoustic-feature sequences into low dimensional latent space. They adopted GPDM as an alternative to HMM, which has been widely used in speech recognition and synthesis. However, it is not easy to apply GPDM to text-to-speech directly because we need another model that converts linguistic contexts into latent space variables. Fernandez *et al.* [19] proposed a framework of f_o contour prediction, in which GP and DNN are combined. In this framework, DNN is used as a context extractor and its hidden-layer values are used as inputs of GP regression. Unlike these frameworks, we consider a more

straightforward framework, in which we predict speech parameters from contexts directly.

C. Relationship Between DNN and GP

Whereas DNNs execute regression using parameters of weight matrices, DGPs do this using pairs of input and output features at each layer. Therefore, DNN and DGP are not directly related to each other. However, it has been shown that a neural network with infinite hidden units and specific prior distribution on a weight matrix is equivalent to a GP [20]. A recent empirical study involving a small amount of training data [21] showed that GP yields similar predictive distribution to a Bayesian neural network without expensive computation processes such as Markov chain Monte Carlo (MCMC).

Some studies proposed transformation approaches between a neural network and kernel methods. For example, the arc-cosine (ArcCos) kernel was proposed [22] as an equivalent kernel to the rectified linear unit (ReLU) activation function. Lee *et al.* generalized transformation from the arbitrary activation function to an equivalent kernel [23]. From the opposite viewpoint, Rahimi and Recht [24] approximated costly kernel regression by inexpensive linear regression in which a finite number of basis functions are used as an approximation of kernel function. Specifically, it was proved that a widely used RBF kernel can be approximated by the inner product of the finite number of cosine function bases. Cutajar *et al.* [8] incorporated this approximation, referred to as random feature expansion, into a DGP, enabling the training of a DGP similar to that of a DNN.

III. STOCHASTIC VARIATIONAL GAUSSIAN PROCESS

This section briefly introduces a single layer GP and GP with stochastic variational inference (SVGP¹) [26], [27]. Let $\mathbf{y} = [y_1, \dots, y_N]^T$ be a sequence of output variables and $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ be input variables of training data, which correspond to a set of frame-level acoustic features and contexts in SPSS, respectively. By introducing latent variables $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]^T$, the marginal likelihood of the output feature sequence \mathbf{y} is given by

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})d\mathbf{f}. \quad (1)$$

When a latent function f is sampled from a GP $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, which consists of a mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, $p(\mathbf{f}|\mathbf{X})$ becomes the following Gaussian distribution:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}; m(\mathbf{X}), K(\mathbf{X}, \mathbf{X})) \quad (2)$$

where $m(\mathbf{X}) = [m(\mathbf{x}_1), \dots, m(\mathbf{x}_N)]^T$, and $K(\mathbf{X}, \mathbf{X})$ is a matrix whose (i, j) element is obtained by the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$. Assuming that the likelihood $p(\mathbf{y}|\mathbf{f})$ is i.i.d., it can be represented as

$$p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^N p(y_i|f(\mathbf{x}_i)). \quad (3)$$

¹We use the term ‘‘SVGP’’ according to the Gaussian process toolkit GPy [25]

The predictive distribution of output y_* given new input \mathbf{x}_* is defined as

$$p(y_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int p(y_*|f_*)p(f_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X})df_* \quad (4)$$

where

$$p(f_*|\mathbf{x}_*, \mathbf{y}, \mathbf{X}) = \int p(f_*|\mathbf{f}, \mathbf{x}_*, \mathbf{X})p(\mathbf{f}|\mathbf{y}, \mathbf{X})d\mathbf{f}. \quad (5)$$

The bottleneck of regression and classification based on GP is computational tractability of the marginal likelihood and predictive distribution. Specifically, training of GP regression requires $\mathcal{O}(N^3)$ computations where N is the number of training data points. Therefore, various approximation techniques have been proposed to reduce computational complexity.

SVGP [26], [27] is an approximation method proposed for an application that uses a large amount of training data. SVGP uses inducing inputs $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_M)$ and corresponding inducing outputs $\mathbf{u} = [f(\mathbf{z}_1), \dots, f(\mathbf{z}_M)]^\top$ in accordance with the sparse GP approach [28]. In general, inducing points can be regarded as the representative points of the training data. Let the number of inducing points M be a much smaller value than the training data samples N , we can reduce the computation complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$. From the definition of GP, the joint distribution of \mathbf{f} and \mathbf{u} is the following Gaussian distribution:

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N} \left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix}; \begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{Z}) \end{bmatrix}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) & K(\mathbf{X}, \mathbf{Z}) \\ K(\mathbf{Z}, \mathbf{X}) & K(\mathbf{Z}, \mathbf{Z}) \end{bmatrix} \right). \quad (6)$$

Hence, the conditional distribution also becomes a Gaussian distribution

$$p(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (7)$$

$$\boldsymbol{\mu} = m(\mathbf{X}) + K(\mathbf{X}, \mathbf{Z})K(\mathbf{Z}, \mathbf{Z})^{-1}(\mathbf{u} - m(\mathbf{Z})) \quad (8)$$

$$\boldsymbol{\Sigma} = K(\mathbf{X}, \mathbf{X}) - K(\mathbf{X}, \mathbf{Z})K(\mathbf{Z}, \mathbf{Z})^{-1}K(\mathbf{Z}, \mathbf{X}). \quad (9)$$

By introducing the inducing points, the marginal likelihood can be written as

$$p(\mathbf{y}) = \iint p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{f}d\mathbf{u}. \quad (10)$$

Let $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})$ be a variational distribution as an approximation of posterior $p(\mathbf{u}|\mathbf{y})$. We obtain the variational lower bound of the log marginal likelihood as follows:

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \left\{ \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})d\mathbf{f} \right\} d\mathbf{u} - \text{KL}(q(\mathbf{u})\|p(\mathbf{u})) \quad (11)$$

where KL represents Kullback-Leibler divergence. Moreover, from Jensen's inequality,

$$\begin{aligned} \log p(\mathbf{y}) &\geq \iint q(\mathbf{u})p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f})d\mathbf{f}d\mathbf{u} - \text{KL}(q(\mathbf{u})\|p(\mathbf{u})) \\ &= \int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f})d\mathbf{f} - \text{KL}(q(\mathbf{u})\|p(\mathbf{u})) \triangleq \mathcal{L} \end{aligned} \quad (12)$$

where $q(\mathbf{f})$ is predictive distribution derived using (7) as follows:

$$\begin{aligned} q(\mathbf{f}) &= \int q(\mathbf{u})p(\mathbf{f}|\mathbf{u})d\mathbf{u} \\ &= \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f) \end{aligned} \quad (13)$$

$$\boldsymbol{\mu}_f = m(\mathbf{X}) + K(\mathbf{X}, \mathbf{Z})K(\mathbf{Z}, \mathbf{Z})^{-1}(\mathbf{m} - m(\mathbf{Z})) \quad (14)$$

$$\begin{aligned} \boldsymbol{\Sigma}_f &= K(\mathbf{X}, \mathbf{X}) - K(\mathbf{X}, \mathbf{Z})K(\mathbf{Z}, \mathbf{Z})^{-1}K(\mathbf{Z}, \mathbf{X}) \\ &\quad + K(\mathbf{X}, \mathbf{Z})K(\mathbf{Z}, \mathbf{Z})^{-1}\mathbf{S}K(\mathbf{Z}, \mathbf{Z})^{-1}K(\mathbf{Z}, \mathbf{X}). \end{aligned} \quad (15)$$

We define the above predictive distribution in (13) as

$$\text{SVGP}(\mathbf{f}; \mathbf{X}, q(\mathbf{u}), \mathbf{Z}) \triangleq \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f) \quad (16)$$

Hence, maximizing the marginal likelihood is approximated by maximizing \mathcal{L} , called evidence lower bound (ELBO). Since ELBO can be decomposed into the sum of respective N training samples as

$$\mathcal{L} = \sum_{i=1}^N \left\{ \mathbb{E}_{q(f(\mathbf{x}_i))} [\log p(y_i|f(\mathbf{x}_i))] - \frac{1}{N} \text{KL}(q(\mathbf{u})\|p(\mathbf{u})) \right\} \quad (17)$$

the ELBO can be maximized by stochastic (or minibatch) gradient descent (ascent) optimization in the same manner as neural network training, which means this framework can be applied to a large amount of training data. The first and second terms of the ELBO represent the fitness of the predictive distribution and the regularization penalty of inducing variables, respectively. The parameters of SVGP are composed of inducing inputs \mathbf{Z} , variational distribution of inducing variables $q(\mathbf{u})$, and mean and kernel function parameters.

The predictive distribution in (4) is transformed into the following expression

$$\begin{aligned} p(y_*|\mathbf{y}) &= \iint p(y_*|f_*)p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f}_*d\mathbf{f} \\ &= \iiint p(y_*|f_*)p(f_*|\mathbf{u})p(\mathbf{u}|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{u}d\mathbf{f}_*d\mathbf{f}. \end{aligned}$$

By introducing the variational distribution, we obtain

$$\begin{aligned} p(y_*|\mathbf{y}) &\approx \iint p(y_*|f_*)p(f_*|\mathbf{u})q(\mathbf{u})d\mathbf{u}d\mathbf{f}_* \\ &= \int p(y_*|f_*)\text{SVGP}(f_*; \mathbf{x}_*, \mathbf{Z}, q(\mathbf{u}))d\mathbf{f}_*. \end{aligned} \quad (18)$$

IV. DEEP GAUSSIAN PROCESSES

DGP [5] is a stacked model of multiple GPs. Specifically, it is assumed with DGPs that the latent function of a single-layer GP is a composite function whose partial functions are sampled from individual GPs. This latent function f is represented as

$$f = f^L \circ f^{L-1} \circ \dots \circ f^1 \quad (19)$$

$$f^\ell = (f^{\ell,1}, \dots, f^{\ell,D_\ell}) \quad (20)$$

$$f^{\ell,d} \sim \mathcal{GP}(m^{\ell,d}(\cdot), k^\ell(\cdot, \cdot)) \quad (21)$$

where L is the number of layers of the stacked model, and D_ℓ is the dimensionality of the ℓ -th layer. A kernel function $k^\ell(\cdot, \cdot)$ is

shared in all dimensions for each layer. Let the output variables of layer ℓ be

$$\begin{aligned} \mathbf{F}^\ell &= [\mathbf{f}^{\ell,1}, \dots, \mathbf{f}^{\ell,D_\ell}] \\ &= \begin{bmatrix} \mathbf{f}_1^{\ell \top} \\ \vdots \\ \mathbf{f}_N^{\ell \top} \end{bmatrix} \\ &= \begin{bmatrix} f_1^{\ell,1} & \dots & f_1^{\ell,D_\ell} \\ \vdots & \ddots & \vdots \\ f_N^{\ell,1} & \dots & f_N^{\ell,D_\ell} \end{bmatrix} \end{aligned} \quad (22)$$

where $f_i^{\ell,d} = f^{\ell,d}(f^{\ell-1}(\dots(f^1(\mathbf{x}_i))))$. The relationship between layers ℓ and $\ell - 1$ is represented using (2) by

$$p(\mathbf{F}^\ell) = \prod_{d=1}^{D_\ell} p(\mathbf{f}^{\ell,d} | \mathbf{F}^{\ell-1}) \quad (23)$$

$$p(\mathbf{f}^{\ell,d} | \mathbf{F}^{\ell-1}) = \mathcal{N}(\mathbf{f}^{\ell,d}; m^{\ell,d}(\mathbf{F}^{\ell-1}), K^{\ell}(\mathbf{F}^{\ell-1}, \mathbf{F}^{\ell-1})). \quad (24)$$

The marginal likelihood of output \mathbf{Y} is given by

$$p(\mathbf{Y} | \mathbf{X}) = \int \dots \int p(\mathbf{Y} | \mathbf{F}^L) p(\mathbf{F}^L | \mathbf{F}^{L-1}) \dots p(\mathbf{F}^1 | \mathbf{X}) d\mathbf{F}^L \dots d\mathbf{F}^1. \quad (25)$$

To use a large amount of training data in DGP, Salimbeni and Deisenroth [9] incorporated SVI into DGP, which is referred to as DSVI-DGP. In this DSVI-DGP, for each layer, they introduced inducing inputs $\mathbf{Z}^\ell = [\mathbf{z}^{\ell,1}, \dots, \mathbf{z}^{\ell,D_{\ell-1}}]$, outputs $\mathbf{U}^\ell = [\mathbf{u}^{\ell,1}, \dots, \mathbf{u}^{\ell,D_\ell}]$, and variational distribution $q(\mathbf{u}^{\ell,d}) = \mathcal{N}(\mathbf{u}^{\ell,d}; \mathbf{m}^{\ell,d}, \mathbf{S}^{\ell,d})$ similarly to SVI. The ELBO is given by

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^N \left\{ \sum_{d=1}^{D_L} \mathbb{E}_{q(\tilde{f}_i^{L,d})} \left[\log p(y_i^d | \tilde{f}_i^{L,d}) \right] \right. \\ &\quad \left. - \frac{1}{N} \sum_{\ell=1}^L \sum_{d=1}^{D_\ell} \text{KL}(q(\mathbf{u}^{\ell,d}) \| p(\mathbf{u}^{\ell,d} | \mathbf{Z}^\ell)) \right\}. \end{aligned} \quad (26)$$

The first and second terms of (26) represent data-fit and complexity penalty, respectively, and $q(\tilde{f}_i^{L,d})$ is a variational posterior defined as

$$q(\tilde{f}_i^{L,d}) = \int \dots \int q(\tilde{f}_i^{L,d} | \mathbf{f}_i^{L-1}) q(\mathbf{f}_i^{L-1} | \mathbf{f}_i^{L-2}) \dots q(\mathbf{f}_i^1 | \mathbf{x}_i) d\mathbf{f}_i^{L-1} \dots d\mathbf{f}_i^1 \quad (27)$$

$$\begin{aligned} q(\mathbf{f}_i^\ell | \mathbf{f}_i^{\ell-1}) &= \prod_{d=1}^{D_\ell} q(f_i^{\ell,d} | \mathbf{f}_i^{\ell-1}) \\ &= \prod_{d=1}^{D_\ell} \text{SVGP}(f_i^{\ell,d}; \mathbf{f}_i^{\ell-1}, \mathbf{Z}^\ell, q(\mathbf{u}^{\ell,d})). \end{aligned} \quad (28)$$

Since the integral of (27) is intractable except particular kernels, it is impossible to compute the exact ELBO. Therefore, we approximate the variational posterior (27) by Monte

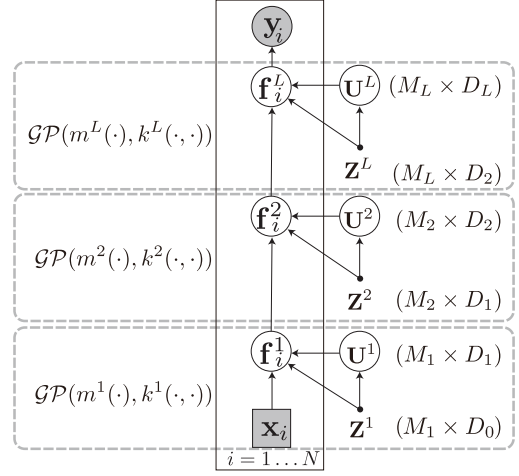


Fig. 1. Graphical representation of deep Gaussian process with doubly stochastic variational inference.

Carlo sampling known as the reparameterization trick [10]. Specifically, instead of marginalizing out hidden-layer variables $\tilde{f}_i^{\ell,d}$, we use a value $\hat{f}_{i,s}^{\ell,d}$ ($\ell = 1, \dots, L$) sampled from $\text{SVGP}(\cdot; \hat{\mathbf{f}}_{i,s}^{\ell-1}, \mathbf{Z}^\ell, q(\mathbf{u}^{\ell,d}))$ where s denotes a sample index and $\hat{\mathbf{f}}_{i,s}^{\ell-1}$ is a sample value in the lower layer. Since the distribution of $\text{SVGP}(\cdot)$ is Gaussian represented by $\mathcal{N}(f_{i,s}^{\ell,d}; \mu_{i,s}^{\ell,d}, (\sigma_{i,s}^{\ell,d})^2)$, the sampling is calculated using the following equation:

$$\hat{f}_{i,s}^{\ell,d} = \mu_{i,s}^{\ell,d} + \epsilon_{i,s}^{\ell,d} \sigma_{i,s}^{\ell,d} \quad (29)$$

where $\epsilon_{i,s}^{\ell,d}$ is a random scalar value sampled from a standard normal distribution.

As a result, the ELBO is approximated by

$$\begin{aligned} \mathcal{L} &\approx \frac{1}{S} \sum_{s=1}^S \sum_{i=1}^N \left\{ \sum_{d=1}^{D_L} \mathbb{E}_{q(\tilde{f}_{i,s}^{L,d})} \left[\log p(y_i^d | \tilde{f}_{i,s}^{L,d}) \right] \right. \\ &\quad \left. - \frac{S}{N} \sum_{\ell=1}^L \sum_{d=1}^{D_\ell} \text{KL}(q(\mathbf{u}^{\ell,d}) \| p(\mathbf{u}^{\ell,d} | \mathbf{Z}^\ell)) \right\} \end{aligned} \quad (30)$$

$$q(\tilde{f}_{i,s}^{L,d}) = \text{SVGP}(\tilde{f}_{i,s}^{L,d}; \hat{\mathbf{f}}_{i,s}^{L-1}, \mathbf{Z}^L, q(\mathbf{u}^{L,d})) \quad (31)$$

where S is the number of Monte Carlo samples.

Since the approximated ELBO can be expressed by the sum of respective training data points and Monte Carlo samples, this approximation is called *doubly* stochastic variational inference (DSVI), and the DGP model with DSVI is trained using stochastic gradient ascent algorithms in the same manner as with SVGP.

The graphical representation of the original DSVI-DGP with $L = 3$ layers is shown in Fig. 1. The model parameters are composed of inducing inputs $\{\mathbf{Z}^\ell\}_{\ell=1}^L$, the variational parameters of inducing outputs $\{\{(\mathbf{m}^{\ell,d}, \mathbf{S}^{\ell,d})\}_{d=1}^{D_\ell}\}_{\ell=1}^L$, and the parameters of mean and kernel functions of Gaussian processes.

V. SPEECH SYNTHESIS BASED ON DEEP GAUSSIAN PROCESS

A. Input and Output Feature

We train two models with different modeling levels: a phone-level model and frame-level model in a similar manner to the Merlin toolkit [29]. When synthesizing speech, the phone and frame level predictions are executed in cascade. First, we predict phone durations from phone-level context using the phone-level model. Next, we make a frame-level context by combining the phone-level context and frame position information obtained from the predicted phone duration. We predict acoustic features from the frame-level context using the frame-level model. For both models, contextual factors include phone-, phrase-, and utterance-level linguistic information with their extended binary features using questions about contextual factors, such as “*Is the previous phoneme a vowel?*” and “*Is the number of moras in the sentence more than 10?*”. In addition to these features, relative position features in the segments are used for the frame-level context.

B. Likelihood Function

To incorporate DGP into SPSS, we must define the likelihood function of acoustic features y given a latent function variable $f(\mathbf{x}_i)$. To evaluate the likelihood $p(y|f(\mathbf{x}))$, we use individual functions for respective acoustic features such as mel-cepstrum, f_o , and voiced/unvoiced (V/UV) flag. In regression cases such as mel-cepstrum, aperiodicity feature, and phone duration, the likelihood is defined by a Gaussian distribution as

$$p(y|f(\mathbf{x})) = \mathcal{N}(y; f(\mathbf{x}), \sigma_v^2) \quad (32)$$

where σ_v^2 is a variance of white noise observed between y and $f(\mathbf{x})$. In the f_o case, we assume that the f_o values in unvoiced regions are single 0-dimensional NULL value. To model the continuous f_o and NULL values simultaneously, we introduce multi-space probability distribution in the same manner as HMM-based speech synthesis [30], which is given by

$$p(y|f(\mathbf{x})) = \begin{cases} \mathcal{N}(y; f(\mathbf{x}), \sigma_v^2) & \text{if voiced} \\ 1 & \text{if unvoiced.} \end{cases} \quad (33)$$

For the V/UV feature represented by a binary value, namely, +1 for voiced or -1 for unvoiced, we use

$$p(y|f(\mathbf{x})) = \Phi(yf(\mathbf{x})) \quad (34)$$

where $\Phi(\cdot)$ is a cumulative distribution function of the standard normal distribution in the same manner as general GP classification [31].

C. Inference and Parameter Generation

When synthesizing speech, we calculate a predictive distribution

$$p(\mathbf{y}_*) = \int p(\mathbf{y}_*|\mathbf{f}_*^L)q(\mathbf{f}_*^L)d\mathbf{f}_*^L. \quad (35)$$

However, the distribution $q(\mathbf{f}_*^L)$ in (27) is intractable. Hence, we approximate $q(\mathbf{f}_*^L)$ by propagation of expectations. Specifically,

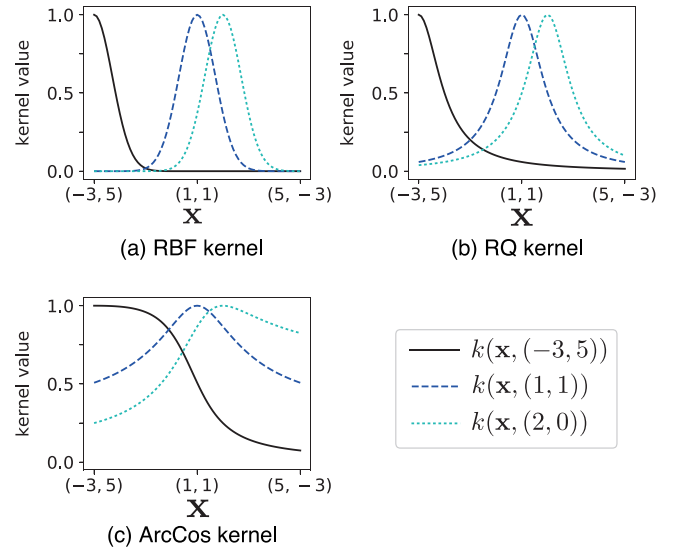


Fig. 2. Example of kernel functions. For the RBF and RQ kernels, $l_1^2 = l_2^2 = 1$ and $\alpha = 1$. For the ArcCos kernel with, we set $P = 1$, $\sigma_{b_0}^2 = \sigma_{b_1}^2 = 1$, and $\sigma_{w_0}^2 = \sigma_{w_1}^2 = 1$.

we regard the predictive mean as an output of each layer as follows

$$\hat{\mathbf{f}}_i^\ell = \mathbb{E}[q(\mathbf{f}_*^\ell | \hat{\mathbf{f}}_*^{\ell-1})] (\ell = 1, \dots, L-1) \quad (36)$$

$$\begin{aligned} q(\mathbf{f}_*^L) &\approx q(\mathbf{f}_*^L | \hat{\mathbf{f}}_*^{L-1}) \\ &= \prod_{d=1}^{D_L} \text{SVGP}(\mathbf{f}_*^{L,d}; \hat{\mathbf{f}}_*^{L-1}, \mathbf{Z}^L, q(\mathbf{u}^{L,d})). \end{aligned} \quad (37)$$

We use predictive mean sequence including dynamic features and predictive variances for parameter generation, which is computed in a similar manner to the study by Tokuda *et al.* [32].

VI. KERNEL FUNCTIONS

We investigate the performance of kernel functions on a DGP-based framework for speech synthesis. Specifically, we use not only a widely used RBF kernel, but also rational quadratic (RQ) kernel and ArcCos kernel. Fig. 2 plots examples of kernel functions whose inputs are on the line between $(-3, 5)$ and $(5, -3)$ in 2-D Euclidean space.

1) *RBF kernel*: An RBF kernel is a typical stationary kernel defined as

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{r^2}{2}\right) \quad (38)$$

$$r = \sqrt{(\mathbf{x} - \mathbf{x}')^\top \mathbf{\Lambda}^{-1} (\mathbf{x} - \mathbf{x}')} \quad (39)$$

$$\mathbf{\Lambda} = \text{diag}[l_1^2, \dots, l_D^2]. \quad (40)$$

As shown in (38), the output of the RBF kernel depends on the distance between two input vectors. Parameters (l_1, \dots, l_D) are called length-scale parameters and used for automatic relevance determination (ARD), which tunes the importance of input dimensions. The disadvantage of the RBF kernel is that

the outputs and gradients tend to be sparse because they become almost zero if two inputs are far from each other. Therefore, it can be expected that the RBF kernel causes a gradient-vanishing problem.

2) *RQ kernel*: An RQ kernel is defined as the sum of infinite RBF kernels and represented by

$$k_{\text{RQ}}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{r^2}{2\alpha}\right)^{-\alpha} \quad (41)$$

where $\alpha (> 0)$ denotes the parameter that forms the contour of the kernel function. For example, the contour becomes gentle when α is small whereas the kernel corresponds to the RBF kernel if $\alpha \rightarrow \infty$. Compared with the RBF kernel, as shown in Fig. 2, the outputs are unlikely to become almost zero even if two inputs are distant.

3) *ArcCos kernel*: An ArcCos kernel [22], [23] is a kernel derived from a neural network with an infinite number of hidden units and the ReLU activation function. It is defined from the following equations.

$$k_0(\mathbf{x}, \mathbf{x}') = \sigma_{b0}^2 + \sigma_{w0}^2 \mathbf{x}^\top \mathbf{\Lambda} \mathbf{x}' \quad (42)$$

for $i = 1 \dots P$

$$k_i(\mathbf{x}, \mathbf{x}') = \sigma_{bi}^2 + \sigma_{wi}^2 \sqrt{k_{i-1}(\mathbf{x}, \mathbf{x})} \sqrt{k_{i-1}(\mathbf{x}', \mathbf{x}')} \cdot (\sin \theta_{i-1} + (\pi - \theta_{i-1}) \cos \theta_{i-1}) \quad (43)$$

$$\theta_{i-1} = \cos^{-1} \frac{k_{i-1}(\mathbf{x}, \mathbf{x}')}{\sqrt{k_{i-1}(\mathbf{x}, \mathbf{x})} \sqrt{k_{i-1}(\mathbf{x}', \mathbf{x}')}} \quad (44)$$

$$k_{\text{ArcCos}}(\mathbf{x}, \mathbf{x}') = \frac{k_P(\mathbf{x}, \mathbf{x}')}{\sqrt{k_P(\mathbf{x}, \mathbf{x})} \sqrt{k_P(\mathbf{x}', \mathbf{x}')}} \quad (45)$$

where P is the number of layers. Although the normalization process in (45) is not used in the original ArcCos kernel, we impose it to restrict the range of kernel outputs for stable computation. Furthermore, we examine the effectiveness of ARD parameter $\mathbf{\Lambda}$, which is not used in the original ArcCos kernel.

Unlike the RBF and RQ kernels, the outputs depend on the positions of two inputs instead of the distance, and it has a sigmoid-like contour as shown in Fig. 2(c). Some studies [22], [33] reported that an SVM using the ArcCos kernel outperformed that using the RBF kernel.

VII. EXPERIMENTS

A. Experimental Conditions

F009, a female speaker, included in the speech synthesis system XIMERA database [34] was used for our experiments. We used 1593 sentences (about 119 minutes) for training data and 60 sentences (about 4.1 minutes) for evaluation data. 60 sentences in the training data were validation data, which were used for hyperparameter tuning and early stopping. In addition to phonetically balanced sentences, the database includes travel-conversation and newspaper-reading sentences. We extracted f_0 , spectral envelope, and aperiodicity from STRAIGHT [35] every 5 ms from the speech signal at a sampling rate of 16 kHz and obtained 0–39th mel-cepstrum, $\log f_0$, and 5-band aperiodicity. We used a 139-dimensional vector consisting of Δ , Δ^2 ,

and V/UV flags as acoustic features. We extracted 243, 82, 136, 76, and 35 contextual factors for phone, mora, accent phrase, breath group, and sentence levels, respectively, using questions as described in Section V-A. By combining the contextual factors and 2-dimensional frame position information, we obtained 574-dimensional input feature vectors. We trained two models: a phone-level model that predicts phone durations and frame-level model that predicts acoustic features. We normalized all input and output variables to zero-mean and unit variance.² The measures of objective evaluation included mel-cepstrum distance [dB], root-mean-square errors (RMSEs) of $\log f_0$ [cent] and phoneme duration [ms], and V/UV error rate [%]. Computation time was evaluated using NVIDIA GeForce GTX980 Ti GPU, Intel Core i9-7900X CPU, and TensorFlow 1.7 implementation.

We investigated various model configurations as shown in Table I. We adopted different types of configurations for the top layer and non-top (i.e., middle and bottom) layers independently because the function of the top layer is regarded as a predictor of acoustic features whereas those of non-top layers are feature extractors of contextual information. We set $l_1 = \dots = l_D = 2$ for the RBF and RQ kernels and $\alpha = 1$ for the RQ kernel as initial parameters of kernel functions. We initialized $l_1 = \dots = l_D = \sqrt{1/D}$ for ArcCos kernel. The number of layers of the ArcCos kernel, P in (43), was set to 3, and the following values were used: $\sigma_{b0} = \dots = \sigma_{b3} = 1$, $\sigma_{w0} = \dots = \sigma_{w3} = 1$. Optimization was done using Adam [12] whose learning rate was 0.01 and minibatch size was 1024. We set the number of Monte Carlo samples S at unity. We stopped optimization after 30 and 300 epochs³ in the frame- and phone-level model training, respectively.

In the same manner as in a previous study [9], we set the following mean functions $m^\ell(\cdot)$ of a DVSI-DGP model to enable us to start optimization. For $\ell = 1$, a mapping function based on primary component analysis was used. In the middle layers $\ell = 2 \dots L - 1$, a copy function $m^\ell(\mathbf{x}) = \mathbf{x}$ was used. The mean function of the top layer was set to $m^L(\mathbf{x}) = \mathbf{0}$ similarly to a typical single-layer GP. These mean functions were not optimized during training whereas parameters of kernel functions were optimized. As initial values of inducing inputs \mathbf{Z}^ℓ , we used the centroids of K -means clustering for each layer, where propagated input features by the mean functions were used. The initial values of variational parameters $\mathbf{m}^{\ell,d}$ and $\mathbf{S}^{\ell,d}$ were a zero vector and identity matrix, respectively. We constrained $\mathbf{S}^{\ell,d}$ to be diagonal, except in the top layer $\ell = L$.

B. Choice of Kernel Function and Number of Layers

We first evaluated the effects of kernel functions and the number of layers on acoustic feature distortions. The model configurations were rr, qq, aa, qr, ar, and a'a', as shown in Table I, whose kernel functions were different from each other. We varied

²In preliminary tests, it was shown that the effect of normalization methods of binary input features on the performance was minute. Therefore, we used zero-mean and unit variance normalization that can be simply implemented.

³Note that there is room for considering the stopping criterion. In preliminary experiments, even if we increased the number of epochs, the errors for validation data hardly increased.

TABLE I
MODEL CONFIGURATIONS FOR EVALUATIONS. WE USE NOTATION $xx(L, D, M)$, WHERE xx REPRESENTS KERNEL TYPE, AND L, D , AND M CORRESPOND TO THE NUMBER OF LAYERS, HIDDEN-LAYER DIMENSIONALITY, AND THE NUMBER OF INDUCING POINTS OF HIDDEN LAYERS, RESPECTIVELY. THE NUMBER OF INDUCING POINTS OF TOP LAYERS WAS 1024

Component / Method		rr	qq	aa	qr	ar	a'a'
Top layer	kernel	RBF	RQ	ArcCos	RQ	ArcCos	ArcCos
	use ARD	Yes	Yes	No	Yes	No	Yes
Mid & bottom layers	kernel	RBF	RQ	ArcCos	RBF	RBF	ArcCos
	use ARD	Yes	Yes	No	Yes	Yes	Yes

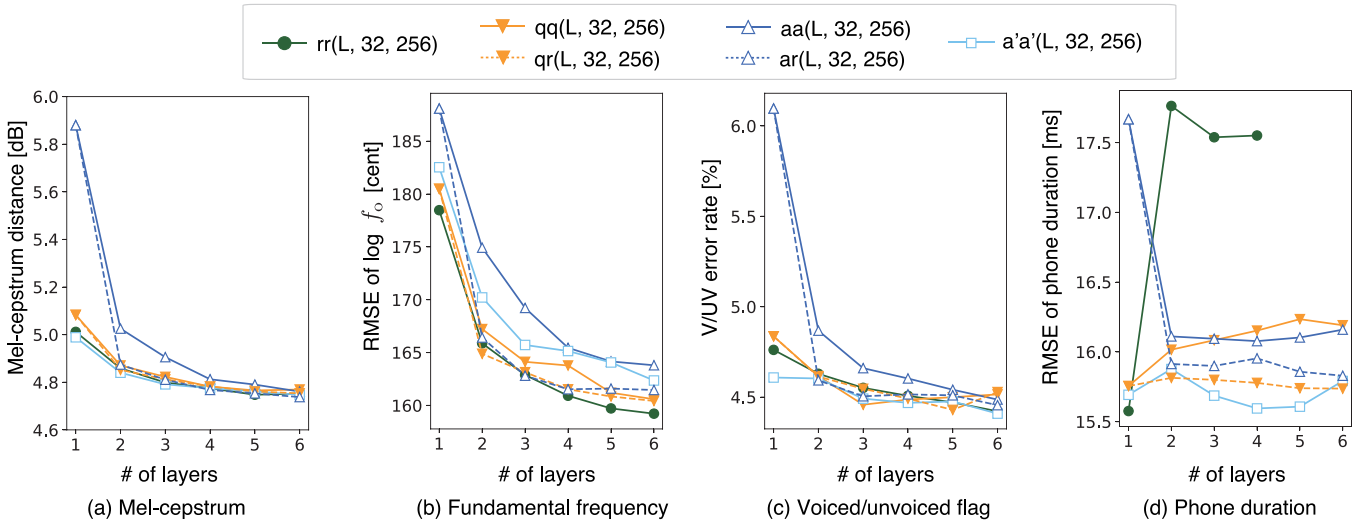


Fig. 3. Acoustic feature distortions as a function of number of DGP layers.

the number of layers from 2 to 6 and also evaluated single-layer GP cases.

Fig. 3 shows the acoustic-feature distortions and errors of mel-cepstrum, $\log f_0$, and phone durations. As shown in Fig. 3(a) and (b), the increase in the number of layers reduced spectral and f_0 distortions. Although the differences among different kernel functions were small, the use of the ArcCos kernel without ARD (aa) had larger distortions than the other kernels.

For duration distortions in Fig. 3(d), the RBF kernel (rr) had a larger distortion than the others. It is noted that the training failed for $L = 5$ and 6, and all output durations became a constant value for any input. One possible reason is the characteristics of the RBF kernel easily causing the gradient vanishing problem. We also see from the result that the effect of changing the number of layers was small, and even a single-layer GP with the RBF or RQ kernels could be sufficient for the duration model.

C. Hidden-Layer Dimensionality

We examined the effect of the dimensionality of hidden-layer variables on the acoustic feature distortions. Too few dimensions lead to a lack in the expressiveness of hidden layers. On the other hand, too many dimensions may make training difficult because Gram matrices generally tend to be sparse for large dimensional input features. Based on the experiments on the original DSVI-DGP study [9], we set $D_\ell = 32$ as a default and varied it from 8 to 256.

Fig. 4 shows the acoustic feature distortions as a function of dimensionality. The model configurations of methods rr, qr and ar are shown in Table I. The training of rr failed when the number of hidden-layer dimensions was large because of vanishing-gradient problem similarly to the previous subsection. Hence, we skip the results of rr in the following experiments. For mel-cepstrum, f_0 , and V/UV flag estimation of qr and ar, low dimensionality, such as 8 and 16, resulted in larger distortions. On the contrary, the phone-duration distortions were less sensitive to the dimensionality. This result could be due to the difference in the vector size of output variables: 139 for frame-level acoustic features and 1 for phone duration.

D. Number of Inducing Points

Inducing points ($\mathbf{Z}^\ell, \mathbf{u}^{\ell,d}$) are supposed to be the pseudo data set that mimics representative points of training data. If the number of inducing points increases to that of training data points, this is equivalent to exact GP regression. However, the inducing points have to be limited because $\mathcal{O}(M^3)$ computation is required when the number of inducing points is M . We varied the number of inducing points of hidden layers, the configurations of which, i.e., qr and ar, are given in Table I.

The results are shown in Fig. 5. The increase in the number of inducing points tends to reduce the distortions of all features. However, the performance got worsened when we used 4096 inducing points. To examine these results in detail, we calculated

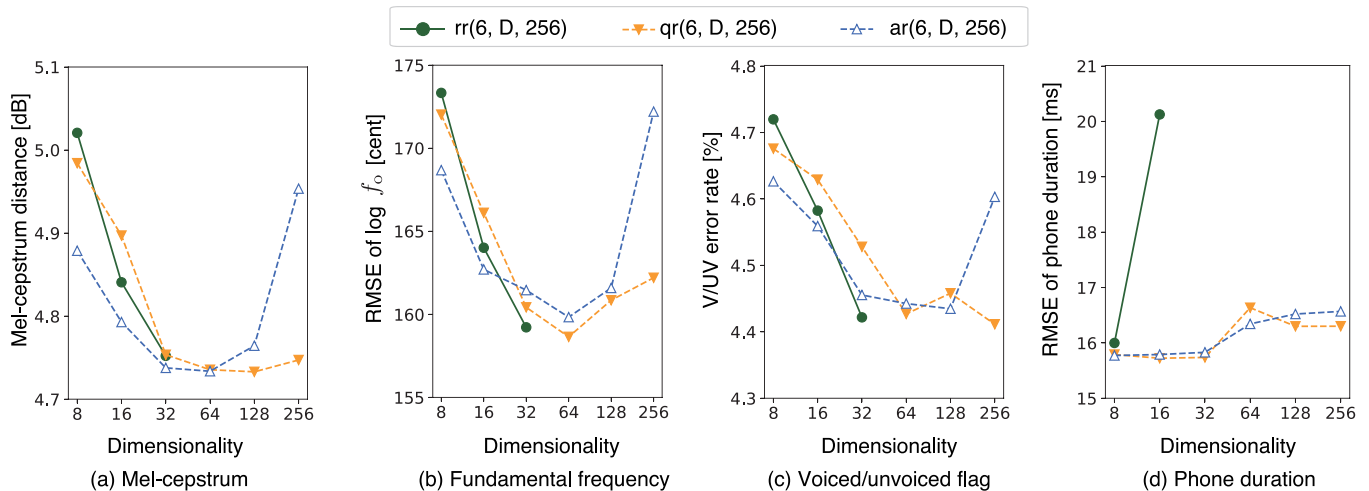


Fig. 4. Acoustic feature distortions as a function of dimensionality of hidden layers.

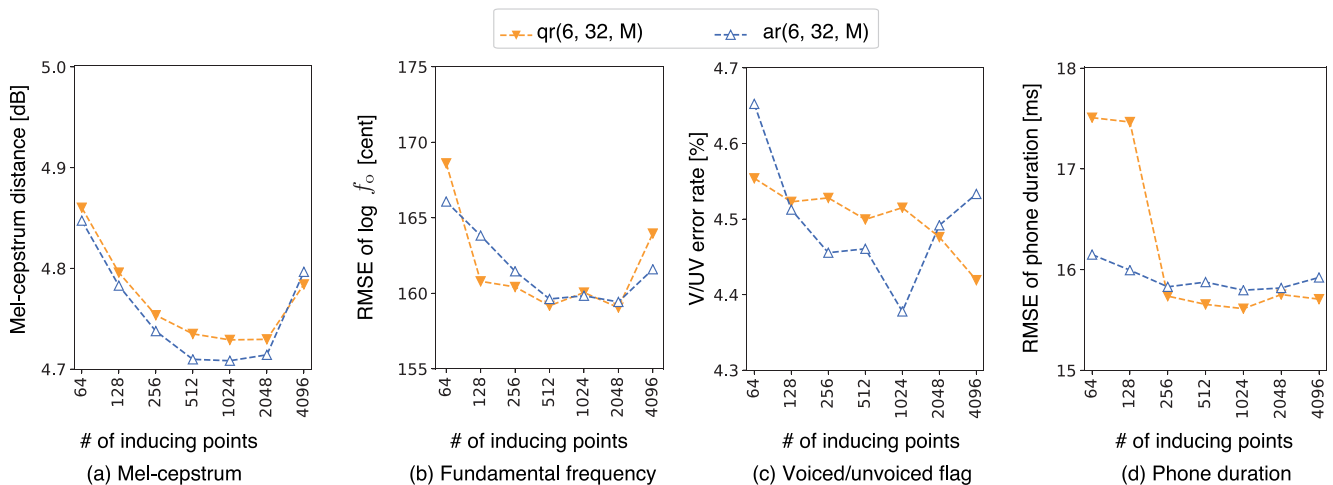


Fig. 5. Acoustic feature distortions as a function of number of inducing points.

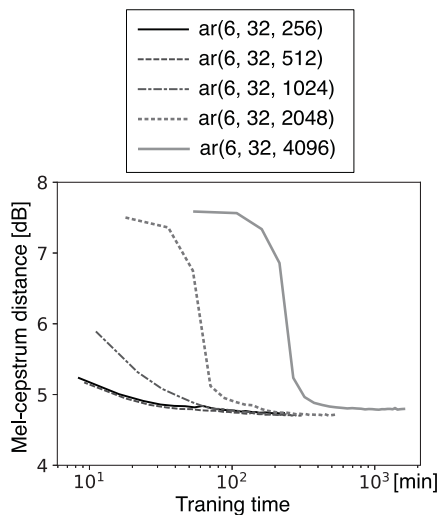


Fig. 6. Mel-cepstrum distortion as a function of computation time in model training using diverse numbers of inducing points of hidden layers.

the epoch-wise mel-cepstrum distortion. Fig. 6 shows the mel-cepstrum distances as a function of the computation time of model training. The case of 4096 inducing points required much time to convergence, and the distortions were larger than the other cases. From these results, too many inducing points caused difficulty in parameter optimization and 1024 inducing points seemed to be sufficient for modeling.

E. Comparison With Other Frameworks

We compared the proposed DGP-based framework with conventional DNN- [1], [36] and PIC-GP-based [13], [14] ones. In the PIC-GP-based framework, the number of inducing points and the maximum number of frames per cluster were both set to 1024, and HMM-based decision tree clustering was used for data partition. We conducted experiments with various combinations of activation functions, hidden unit sizes, and numbers of layers for the DNN-based framework. We used the ReLU and hyperbolic tangent (tanh) for activation functions and varied the

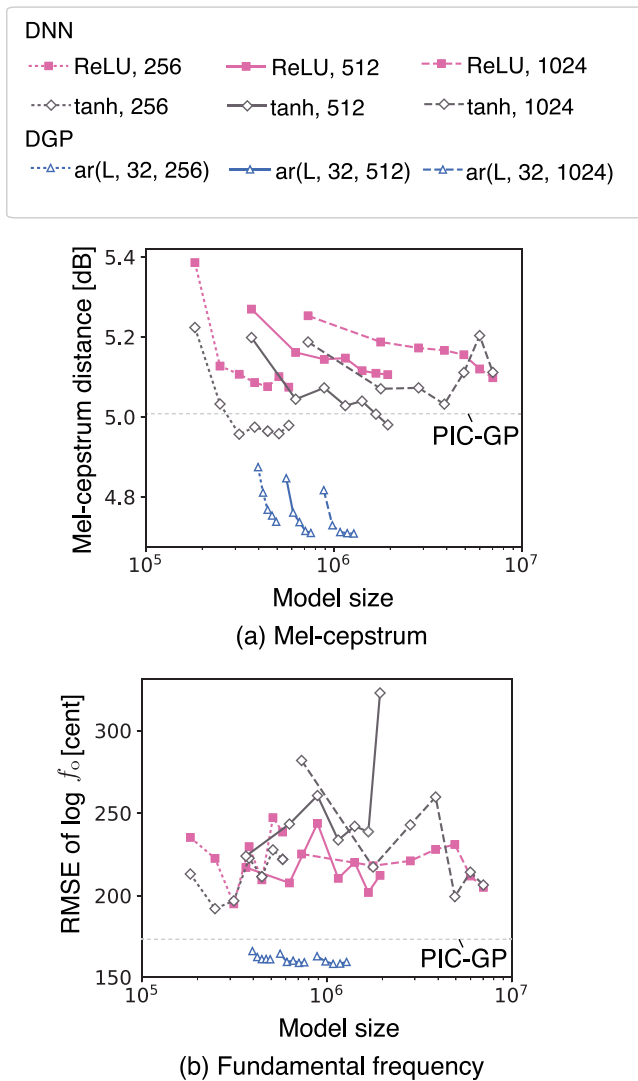


Fig. 7. Comparison of proposed DGP-, feed-forward DNN-, and PIC-GP-based frameworks in terms of acoustic feature distortions. ReLU and tanh are activation functions and 256, 512, and 1024 are the numbers of hidden units for DNN-based frameworks. Model configurations of proposed DGP-based framework are shown in Table I and 256, 512, and 1024 of proposed DGP-based framework correspond to the numbers of inducing points. The dashed horizontal line represents results of PIC-GP-based framework. Points on same line differ by the number of layers.

number of hidden layers of DNNs from 1 to 7. The learning rate for Adam was 0.01 and weight decay coefficient 1.0×10^{-6} was used for regularization. The number of layers of the proposed DSVI-DGP models changed from 2 to 6.

Fig. 7 shows the mel-cepstrum distances and f_0 RMSEs as a function of model size. Even the 2-layer model configuration of the proposed DGP-based framework yielded smaller distortions than those of the feed-forward DNN- and PIC-GP-based frameworks. The model configurations of DGP-based framework did not increase f_0 distortions with the increase in the numbers of layers, unlike with the DNN-based framework. Therefore, the proposed DGP-based framework provides predictable results

and may reduce laborious tuning of model setting compared with a DNN-based framework.

F. Hyperparameter Tuning

We conducted an evaluation using automatic hyperparameter tuning based on Bayesian optimization (BO) [37] to compare the proposed DGP-based framework with a DNN-based one. We optimized regularization parameters, such as weight decay and dropout rate. Tables II and III list the hyperparameters tuned for the DNN and DGP models, respectively. The minibatch size was fixed to 1024, and the maximum epochs in training were 30. For the BO components, expected improvement (EI) [37] was used as an acquisition function, and the RBF kernel was used as a kernel function. As a target function of BO, we used mean squared error for the validation data set. We first trained models using ten randomly chosen hyperparameter sets as warm-up data. We then ran 100 and 50 BO iterations for the DNN and DGP hyperparameter tunings, respectively, both of which took approximately eight days.

The best hyperparameters found by BO are shown in Tables II and III and the acoustic feature distortions are shown in Table IV. From the DNN model results that hyperparameter tuning using BO reduced the distortions compared with those shown in Fig. 7, and the DNN model with optimized hyperparameters resulted in a smaller duration distortion than the DGP model. However, the distortions of mel-cepstrum, f_0 , and V/UV using the DNN model were still larger than those using the DGP model.

We also compared the inference computational complexities of DNN and DGP. The real-time factors of inference by the DNN and DGP models were approximately 0.070 and 0.079 on average.

G. Multiple Data Amounts

We evaluated the performance of multiple amounts of training data to confirm the advantage of our DGP-based framework, which is invulnerable to overfitting. We created smaller-sized subsets using 20%, 40%, 60%, and 80% of the original training data. Regarding the model configurations of the DNN and DGP models, we used the hyperparameters obtained by the BO mentioned in Section VII-F. The maximum number of epochs during training were 100. The results are shown in Fig. 8. The mel-cepstrum and f_0 distortions using the DGP model were smaller than those using the DNN model even when the training data amounts were small. Moreover, we see that the differences of $\log f_0$ RMSEs were consistently around 10 cent between the DNN and DGP models for all data subsets.

H. Subjective Evaluations

To evaluate the perceptual quality of synthetic speech, we conducted crowdsourcing-based subjective evaluations by using mean opinion score (MOS) and preference tests.⁴ More specifically, these tests were conducted using webMUSHRA

⁴Speech samples are available at: <http://www.kbys.ip.titech.ac.jp/demo/dgpps/> or <https://github.com/hyama5/DeepGPTSSamples/>

TABLE II
HYPERPARAMETER TUNING CANDIATES AND BEST HYPERPARAMETERS FOR DNN OBTAINED BY BAYESIAN OPTIMIZATION

Parameter	Candidate	Type	Modeling model	
			Frame	Phone
Num of layers	{1, 2, 3, 4, 5, 6, 7}	Discrete	3	1
Activation	{ReLU, tanh}	Categorical	ReLU	ReLU
Num of hidden units	{128, 256, 512, 1024, 2048}	Discrete	2048	1024
Dropout rate	[0, 0.5]	Continuous	0.5	0
Weight decay	$[10^{-10}, 10^{-1}]$	Continuous	1.97×10^{-6}	5.12×10^{-4}
Learning rate	$[10^{-4}, 10^{-1}]$	Continuous	10^{-4}	10^{-4}

TABLE III
HYPERPARAMETER TUNING CANDIATES AND BEST HYPERPARAMETERS FOR DGP OBTAINED BY BAYESIAN OPTIMIZATION

Parameter	Candidate	Type	Modeling model	
			Frame	Phone
Num of layers	{1, 2, 3, 4, 5, 6}	Discrete	6	4
Top layer	Kernel	{RBF, RQ, ArcCos}	ArcCos	ArcCos
	Use ARD	{Yes, No}	No	Yes
	Num of inducing points	{64, 128, 256, 512, 1024}	Discrete	64
Middle and bottom layers	Kernel	{RBF, RQ, ArcCos}	ArcCos	ArcCos
	Use ARD	{Yes, No}	Yes	Yes
	Num of inducing points	{64, 128, 256, 512, 1024}	Discrete	1024
Dimensionality	{8, 16, 32, 64, 128}	Discrete	64	32
Learning rate	$[10^{-4}, 10^{-1}]$	Continuous	4.21×10^{-2}	9.72×10^{-3}

TABLE IV
ACOUSTIC FEATURE DISTORTIONS OF DNN AND DGP USING BEST HYPERPARAMETERS OBTAINED BY BAYESIAN OPTIMIZATION

Measures	DNN	DGP
Mel-cepstrum distance [dB]	4.82	4.69
RMSE of $\log f_o$ [cent]	166	157
V/UV error rate [%]	4.52	4.40
RMSE of phone duration [ms]	15.6	16.2

[38], a web-based listening test framework. Participants were asked to listen to speech samples using headphones. They could listen to the samples as many times as they required and choose their answers on a web browser. We used the model configurations obtained by BO shown in Section VII-F. We also evaluated the PIC-GP-based framework described in the previous experiment. We used vocoded speech samples (VOC), which were re-synthesized from extracted acoustic features, and original recordings (ORIG) as the ground truth. When synthesizing vocoded speech, we smoothed the extracted log spectrogram and aperiodicity using a 3-frame moving average filter in the time domain. This is because breathy noise was perceived on the VOC with a non-smoothed STRAIGHT spectrogram, which lowered the subjective scores.

In the MOS test, 75 participants listened to the synthetic speech samples and rated the naturalness of them on a five-point scale: 5: excellent, 4: good, 3: fair, 2: poor, and 1: bad. Four sentences were randomly chosen from the 60 sentences of evaluation data for each participant. Fig. 9 shows the results. The proposed DGP-based framework significantly outperformed the DNN- and PIC-GP-based ones. Furthermore, the DGP-based framework gave a comparable score with the VOC.

In the preference test, 30 participants evaluated pairs of speech samples. Ten sentences were randomly chosen from the 60 sentences of the evaluation data for each test and participant. Each participant listened to the pairs of speech samples and selected the most natural one. The results are listed in Table V. The proposed DGP-based framework resulted in higher scores than the DNN and VOC, and the difference between the VOC and DGP was smaller than that between the VOC and DNN.

VIII. DISCUSSION

In the previous section, we compared the proposed DGP-based framework with the feed-forward DNN-based one, both of which perform frame-level modeling and prediction. Although using the feed-forward DNN-based framework is the basic one, state-of-the-art speech synthesis studies have various extensions beside the feed-forward structure. Therefore, in this section, we describe future directions of DGP-based speech synthesis.

LSTM-RNN-based speech synthesis [2] is a useful alternative to DNN-based speech synthesis, in which we can model temporal information. To evaluate the performance of an LSTM-RNN-based framework, we trained an LSTM-RNN model consisting of 2 feed-forward and 2 Bi-LSTM layers with 1024 hidden units based on the study by Fan *et al.* [2]. We used the same data set as that used in the previous section, and observed 4.68 dB in mel-cepstrum distance and 178 cent in $\log f_o$ RMSE. The mel-cepstrum distance was 0.14 dB lower than that using the DNN-based framework and comparable with the proposed DGP-based framework. We also conducted a preference test to evaluate the DGP- and LSTM-RNN-based frameworks using the same condition as the preference test discussed in Section VII-H. The results are listed in Table VI. Although the proposed DGP-based

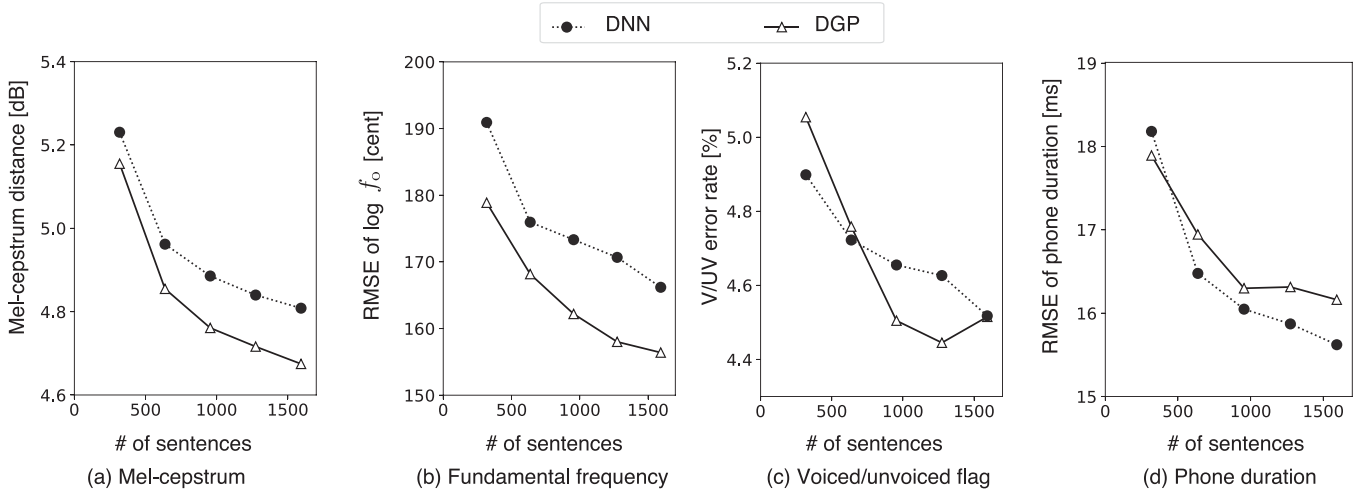


Fig. 8. Acoustic feature distortions as a function of amount of training data.

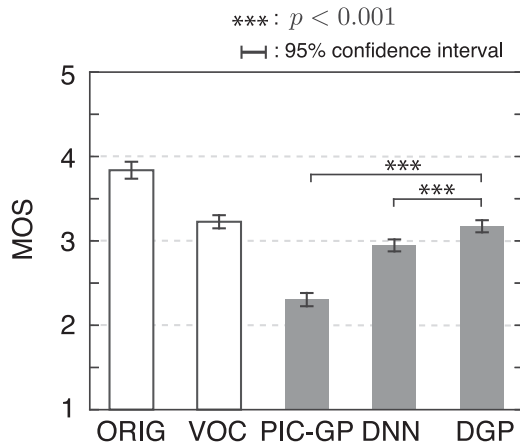


Fig. 9. Subjective evaluation using MOS test.

TABLE V
SUBJECTIVE EVALUATION USING PREFERENCE TEST

VOC	DNN	DGP	p -value	$ Z $ -score
63.3	36.7		$< 10^{-4}$	5.18
54.7		45.3	$< 10^{-4}$	4.79
	35.7	64.3	0.104	1.62

TABLE VI
SUBJECTIVE EVALUATION USING PREFERENCE TEST COMPARING
LSTM-RNN-BASED SPEECH SYNTHESIS

LSTM-RNN	DGP	p -value	$ Z $ -score
44.3	55.7	$< 10^{-4}$	5.18

framework does not have a recurrent structure, we see that it outperformed the LSTM-RNN-based one. Thus, we can expect that the performance of the DGP-based framework will improve using a recurrent architecture.

When we incorporate a recurrent structure into a DGP, we should take the computational complexity into account. DGP

inference takes longer than DNN inference because the computation of Gram matrices is required for calculating a DGP, while a DNN only requires simple linear transformation for each layer.

IX. CONCLUSIONS

We proposed a DSVI-DGP-based framework for speech synthesis, which is based on a probabilistic model with stacked Bayesian kernel regression. The proposed framework uses DSVI, which allows training with stochastic optimization for large-scale training data and the use of arbitrary kernel functions. Different from DNN, DGP is less vulnerable to overfitting because the training of a DGP model is based on the maximization of marginal likelihood. From the experimental evaluations, the proposed DGP-based framework outperformed the feed-forward DNN- and PIC-GP-based frameworks. We also conducted experiments with various model configurations and showed that the increase in layers and inducing points tended to reduce mel-cepstrum and f_0 distortions. We also found that the best kernel function and dimensionality of hidden layers depended on the output features. For future work, we will expand the DSVI-DGP into recurrent, convolution, and attention-based structure, used in state-of-the-art neural-network-based speech synthesis systems instead of a feed-forward DNN-based framework.

REFERENCES

- [1] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 7962–7966.
- [2] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1964–1968.
- [3] J. Sotelo *et al.*, "Char2wav: End-to-end speech synthesis," in *Proc. Int. Conf. Learn. Represent.*, 2017.
- [4] Y. Wang *et al.*, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2017, pp. 4006–4010.
- [5] A. Damianou and N. Lawrence, "Deep Gaussian processes," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2013, pp. 207–215.

- [6] Z. Dai, A. Damianou, J. González, and N. Lawrence, "Variational auto-encoded deep Gaussians processes," in *Proc. Int. Conf. Learn. Represent.*, 2016.
- [7] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, and R. Turner, "Deep Gaussian processes for regression using approximate expectation propagation," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1472–1481.
- [8] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone, "Random feature expansions for deep Gaussian processes," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 884–893.
- [9] H. Salimbeni and M. Deisenroth, "Doubly stochastic variational inference for deep Gaussian processes," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4591–4602.
- [10] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Represent.*, 2014.
- [11] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.
- [12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015.
- [13] T. Koriyama, T. Nose, and T. Kobayashi, "Statistical parametric speech synthesis based on Gaussian process regression," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 2, pp. 173–183, Apr. 2014.
- [14] T. Koriyama and T. Kobayashi, "Prosody generation using frame-based Gaussian process regression and classification for statistical parametric speech synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 4929–4933.
- [15] E. Snelson and Z. Ghahramani, "Local and global sparse Gaussian process approximations," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2007, pp. 524–531.
- [16] T. Koriyama and T. Kobayashi, "A comparison of speech synthesis systems based on GPR, HMM, and DNN with a small amount of training data," in *Proc. Annu. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 3496–3500.
- [17] D. Moungrsri, T. Koriyama, and T. Kobayashi, "GPR-based Thai speech synthesis using multi-level duration prediction," *Speech Commun.*, vol. 99, pp. 114–123, 2018.
- [18] G. E. Henter, M. R. Freaan, and W. B. Kleijn, "Gaussian process dynamical models for nonparametric speech representation and synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2012, pp. 4505–4508.
- [19] R. Fernandez, A. Rendel, B. Ramabhadran, and R. Hoory, "F0 contour prediction with a deep belief network-Gaussian process hybrid model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 6885–6889.
- [20] R. M. Neal, "Priors for infinite networks," in *Bayesian Learning for Neural Networks*. New York, NY, USA: Springer, 1996, pp. 29–53.
- [21] A. G. de G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani, "Gaussian process behaviour in wide deep neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018.
- [22] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 342–350.
- [23] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as Gaussian processes," in *Proc. Int. Conf. Learn. Represent.*, 2018.
- [24] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 1177–1184.
- [25] GPy, "GPy: A Gaussian process framework in python," 2012. [Online]. Available: <http://github.com/SheffieldML/GPy>
- [26] J. Hensman, N. Fusi, and N. Lawrence, "Gaussian processes for big data," in *Proc. 29th Conf. Uncertainty Artif. Intell.*, 2013, pp. 282–290.
- [27] J. Hensman, A. G. de G. Matthews, and Z. Ghahramani, "Scalable variational Gaussian process classification," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2015, pp. 1648–1656.
- [28] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2006, pp. 1257–1264.
- [29] Z. Wu, O. Watts, and S. King, "Merlin: An open source neural network speech synthesis system," in *Proc. 9th ISCA Speech Synthesis Workshop*, 2016.
- [30] K. Tokuda, T. Masuko, N. Miyazaki, and T. Kobayashi, "Multi-space probability distribution HMM," *IEICE Trans. Inf. Syst.*, vol. E85-D, no. 3, pp. 455–464, 2002.
- [31] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
- [32] K. Tokuda, T. Kobayashi, and S. Imai, "Speech parameter generation from HMM using dynamic features," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1995, pp. 660–663.
- [33] C. C. Cheng and B. Kingsbury, "Arccosine kernels: Acoustic modeling with infinite neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 5200–5203.
- [34] H. Kawai, T. Toda, J. Ni, M. Tsuzaki, and K. Tokuda, "XIMERA: A new TTS from ATR based on corpus-based technologies," in *5th ISCA Workshop Speech Synthesis*, 2004, pp. 179–184.
- [35] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigne, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds," *Speech Commun.*, vol. 27, nos. 3/4, pp. 187–207, 1999.
- [36] Y. Qian, Y. Fan, W. Hu, and F. K. Soong, "On the training aspects of deep neural network (DNN) for parametric TTS synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 3829–3833.
- [37] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *J. Global Optim.*, vol. 21, no. 4, pp. 345–383, 2001.
- [38] M. Schoeffler *et al.*, "webMUSHRA—A comprehensive framework for Web-based listening tests," *J. Open Res. Softw.*, vol. 6, no. 1, p. 8, 2018.



Tomoki Koriyama (M'13) received the B.E. degree in computer science in 2009 and the M.E. and Dr.Eng. degrees in information processing in 2010 and 2013, respectively, all from the Tokyo Institute of Technology, Tokyo, Japan.

In 2013, he joined the Research Laboratory of Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, as a Japan Society for the Promotion of Science Research Fellow. In 2014, he became an Assistant Professor with the Interdisciplinary Graduate School of Science and

Engineering, Tokyo Institute of Technology. He is currently with the School of Engineering, Tokyo Institute of Technology, Yokohama, Japan.

Dr. Koriyama was a recipient of The Awaya Prize Young Researcher Award from the Acoustic Society of Japan. He is a member of the International Speech Communication Association, the Acoustical Society of Japan, the Institute of Electronics, Information and Communication Engineers, and the Information Processing Society of Japan.



Takao Kobayashi (M'82–SM'04–F'17) received the B.E. degree in electrical engineering in 1977 and the M.E. and Dr.Eng. degrees in information processing in 1979 and 1982, respectively, all from the Tokyo Institute of Technology, Tokyo, Japan.

In 1982, he joined the Research Laboratory of Precision Machinery and Electronics, Tokyo Institute of Technology as a Research Associate. From 1989 to 1998, he was an Associate Professor in the same Laboratory. In 1998, he became a Professor with the Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology. He is currently with the School of Engineering, Tokyo Institute of Technology, Yokohama, Japan.

Dr. Kobayashi was a corecipient of both the Best Paper Award and the Inose Award from the Institute of Electronics, Information and Communication Engineers (IEICE) in 2001, and the TELECOM System Technology Prize from the Telecommunications Advancement Foundation Award, Japan, in 2001 and 2008. He was also a recipient of the IEICE Information and Systems Society Distinguished Service Award in 2010. He served as the Chair of the Speech Committee of the IEICE and Acoustical Society of Japan (ASJ) in 2007 and 2008. He is a Fellow of the IEICE and a member of the International Speech Communication Association, ASJ, and the Information Processing Society of Japan.

Dr. Kobayashi was a corecipient of both the Best Paper Award and the Inose Award from the Institute of Electronics, Information and Communication Engineers (IEICE) in 2001, and the TELECOM System Technology Prize from the Telecommunications Advancement Foundation Award, Japan, in 2001 and 2008. He was also a recipient of the IEICE Information and Systems Society Distinguished Service Award in 2010. He served as the Chair of the Speech Committee of the IEICE and Acoustical Society of Japan (ASJ) in 2007 and 2008. He is a Fellow of the IEICE and a member of the International Speech Communication Association, ASJ, and the Information Processing Society of Japan.