Interspeech 2018
2-6 September 2018, Hyderabad

Gaussian Process Neural Networks for Speech Recognition

Max W. Y. Lam[1]*, *Shoukang Hu*[1]*, *Xurong Xie*[2], *Shansong Liu*[1],
Jianwei Yu[1], *Rongfeng Su*[3], *Xunying Liu*[1], *Helen Meng*[1]

[1]Department of Systems Engineering and Engineering Management,
[2]Department of Electronic Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong SAR, China
[3]Key Laboratory of Human-Machine Intelligence-Synergy Systems,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

wylam, skhu, xyliu@se.cuhk.edu.hk

Abstract

Deep neural networks (DNNs) play an important role in state-of-the-art speech recognition systems. One important issue associated with DNNs and artificial neural networks in general is the selection of suitable model structures, for example, the form of hidden node activation functions to use. Due to lack of automatic model selection techniques, the choice of activation functions has been largely empirically based. In addition, the use of deterministic, fixed-point parameter estimates is prone to over-fitting when given limited training data. In order to model both models structural and parametric uncertainty, a novel form of DNN architecture using non-parametric activation functions based on Gaussian process (GP), Gaussian process neural networks (GPNN), is proposed in this paper. Initial experiments conducted on the ARPA Resource Management task suggest that the proposed GPNN acoustic models outperformed the baseline sigmoid activation based DNN by 3.40% to 24.25% relatively in terms of word error rate. Consistent performance improvements over the DNN baseline were also obtained by varying the number of hidden nodes and the number of spectral basis functions.

Index Terms: speech recognition, deep neural networks, Gaussian process, variational inference

1. Introduction

Recent advances in speech recognition technologies over a wide range of applications demonstrate that deep neural networks (DNNs) are indispensable in achieving state-of-the-art performances [1]. The works of Han et al. [2] and Saon et al. [3] suggested that deep learning based acoustic models can achieve human-competitive performance on conversational telephone speech recognition. Despite their successful and increasingly wider application in speech processing applications, several fundamental and highly challenging problems associated with DNNs remain to be solved. First, the use of deterministic, fixed-point model update is unable to consider parametric uncertainty. This can lead to over-fitting when only limited amounts of training data is available. More importantly, due to the lack of principled model selection method, the choice of DNN model architecture, for example, represented by the form of activation functions, has been largely empirically based. For such problems, Abdelaziz et al. [4] studied the propagation of observational uncertainties through DNNs. This paper is distinct in that we address both parametric and structural uncertainty using a novel, generalized form of non-parametric activation functions derived from Gaussian process (GP) when designing DNN-based acoustic models for speech recognition systems.

A series of previous works have been done to model parametric uncertainty in DNNs. In 1992, MacKay [5] formulated DNNs in a Bayesian fashion by assuming random Gaussian weights, resulting in a popular class of Bayesian neural networks. Soon after MacKay's work, Neal [6] proved that single-hidden-layer Bayesian neural networks, in the limit of infinite width, is equivalent to Gaussian processes. More recently, Hazan and Jaakkola [7] discussed constructing a kind of kernel functions in GPs such that it is mathematically equivalent to infinitely wide deep neural networks, yet their construction does not go beyond two hidden layers. Taking a similar direction, Lee et al. [8] proposed NNGP, a more efficient way to approximate deep, infinite-width neural networks as GPs. They showed that this kind of GPs can outperform deep, finite-width neural networks due to GPs' anti-overfitting property. However, we conceive that GPs still cannot substitute DNNs since in DNNs there are useful techniques that are likely not convertible to GPs, such as the use of high ways [9, 10], or batch normalisation trick [11]. Therefore, rather than approximating DNNs in GPs, this paper employs GP in DNNs to model non-deterministic activation functions. Our work is inspired by the seminal work of deep Gaussian processes (DGPs) [12] where activation functions in a Bayesian neural network are all replaced by Gaussian processes.

In our work, activation functions modelled by GPs are used to address a form of structural uncertainty – uncertain choice of activation functions. To gain generalizability, we make use of generalized spectral kernels [13], which have been proven capable of approximating any continuous, bouneded kernels. From generalized spectral kernels, we derive generalized activation functions. Having defined the distribution of activation functions, we marginalize over all random function parameters as a means of trying all possible activation functions. By doing so, our proposed Gaussian process neural networks (GPNNs) can effectively handle both parametric and structural uncertainty.

In this paper, we directly compare the performance of GPNNs with DNNs by replacing the first hidden layer as a GPNN layer on the ARPA Resource Management (RM) task. In the experiments, we vary the number of hidden nodes in a large range to demonstrate GPNNs' ability to reduce the risk of over-fitting and under-fitting. Also, we evaluate the performance of GPNNs with different numbers of spectral basis functions.

*Both authors contributed equally to this work. This research was supported by MSRA grant no. 6904412 and Chinese University of Hong Kong (CUHK) grant no. 4055065.

10.21437/Interspeech.2018-1823

2. Background

In this section, we formally define our notations and provide some background of deep neural networks and Gaussian process that are necessary to understand the rest of this paper.

2.1. Standard form of deep neural networks

We begin by defining a standard form of deep neural networks (DNNs) as a multi-dimensional mapping:

$$f_{\text{DNN}} : \mathbb{R}^{D_{\text{in}}} \mapsto \mathbb{R}^{D_{\text{out}}}, \tag{1}$$

where D_{in} and D_{out} denote the dimension of input data and the dimension of output data, respectively. In DNNs, there are three types of layers: (1) input layer $\mathbf{x} \in \mathbb{R}^{D_{\text{in}}}$, (2) hidden layers $f_h : \mathbb{R}^{D_{h-1}} \mapsto \mathbb{R}^{D_h}$ for $h = 1, \dots, H$ where $D_0 \triangleq D_{\text{in}}$, and (3) output layer $f_{\text{out}} : \mathbb{R}^{D_H} \mapsto \mathbb{R}^{D_{\text{out}}}$. Following this notation, we can specify the layer widths of a standard DNN simply by a list of $H + 2$ integers, i.e., $(D_{\text{in}}, D_1, \dots, D_H, D_{\text{out}})$, where H typically denotes the number of hidden layers. To forwardly pass an input x through the layers, i.e., $f_{\text{DNN}}(\mathbf{x}) = f_{\text{out}}(\cdots f_2(f_1(\mathbf{x}))\cdots)$, we recursively calculate, for $h = 1, \dots, H$,

$$\mathbf{z}_h = \mathbf{W}_h \bullet f_{h-1}(\mathbf{x}) + \mathbf{b}_h, \;\; f_h(\mathbf{x}) = \sigma(\mathbf{z}_h), \tag{2}$$

where $\bullet$ is the notation of dot product operation, $\mathbf{z}_i^h$ is the ith output of an affine mapping applied to the outputs of previous layer with optimizable parameters $\mathbf{W}_h \in \mathbb{R}^{D_{h+1} \times D_h}$ and $\mathbf{b}_h \in \mathbb{R}^{D_{h+1}}$, $f_0(\mathbf{x}) \triangleq x$ is the initialization of recursion, and for $h = 1, \dots, H$, $f_h(\mathbf{x})$ is produced after passing through an activation function $\sigma(\cdot)$. Notably, in practice $\sigma(\cdot)$ is usually chosen from some common functions, e.g. *sigmoid*, *tanh* and *ReLU*, that are proven to be *universal* [14]. After obtaining $f_h(\mathbf{x})$, the outputs of DNNs can be computed by going through an normalized affine mapping yet without activation:

$$f_{\text{DNN}}(\mathbf{x}) = \frac{1}{\sqrt{D_H}} \mathbf{W}_{\text{out}} f_H(\mathbf{x}) + \mathbf{b}_{\text{out}}, \tag{3}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{D_{\text{out}} \times D_H}$ and $\mathbf{b}_{\text{out}} \in \mathbb{R}^{D_{\text{out}}}$.

2.2. Gaussian process

A Gaussian process (GP) [15] is random process such that any finite collection of random variables in Gaussian process follows a multivariate Gaussian distribution. If a function follows Gaussian process, it is represented as

$$f(\mathbf{z}) \sim \mathcal{GP}\left(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}')\right), \tag{4}$$

where $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^{D_{\text{in}}}$ are arbitrary inputs, $m(\cdot)$ is the mean function and $k(\cdot, \cdot)$ is the kernel function. In fact, there is a tight connection between deep neural networks and Gaussian processes [8] – if $\mathbf{W}_{ij}^{\text{out}}$ and $\mathbf{b}_i^{\text{out}}$ are taken to be identically and independently distributed (*i.i.d.*), in the limit of infinite width of the output layer ($D_H \to \infty$), it follows from the Central Limit Theorem that $f_{\text{DNN}}(\mathbf{x})$ is a normalized sum of *i.i.d.* random variables that tends toward a multivariate Gaussian distribution. Thus, any finite collection of $\{f_{\text{DNN}}(\mathbf{x}_1), \dots, f_{\text{DNN}}(\mathbf{x}_k)\}$ will be multivariate Gaussian distributed, which resembles a GP.

In GP regression, both the mean function and the kernel function need to be specified before training. In practice, mean function is typically set to zero by assuming that the distribution of outputs has been well transformed [15]. On the other hand, kernel function has been extensively studied [16], as it controls the curvature properties of the its realization function $f(\cdot)$, such as the degree of smoothness and periodicity. Any function satisfying positive definiteness, i.e., $\sum_{i,j} c_i c_j k(\mathbf{z}_i, \mathbf{z}_j) > 0$ for all $c_i, c_j \in \mathbb{R}$ and $\mathbf{z}_i, \mathbf{z}_j \in \mathbb{R}^{D_{\text{in}}}$ is a valid kernel function. Not surprizingly, a number of kernel functions exist in the literature. In this paper, we employ a form of kernel functions that generalizes over all non-stationary processes, called *generalized spectral kernels* [17], which takes the form of

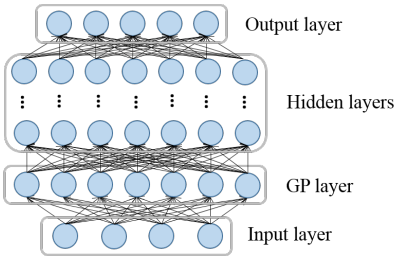

Figure 1: *Architecture of proposed GPNN*

$$k(\mathbf{z}, \mathbf{z}') = \phi(\mathbf{z})^T \phi(\mathbf{z}'), \tag{5}$$

$$\phi(\mathbf{z}) = \frac{1}{\sqrt{S}} \begin{bmatrix} \cos(\boldsymbol{\omega}_1 \bullet \mathbf{z}) + \cos(\boldsymbol{\omega}_1' \bullet \mathbf{z}) \\ \sin(\boldsymbol{\omega}_1 \bullet \mathbf{z}) + \sin(\boldsymbol{\omega}_1' \bullet \mathbf{z}) \\ \vdots \\ \cos(\boldsymbol{\omega}_S \bullet \mathbf{z}) + \cos(\boldsymbol{\omega}_S' \bullet \mathbf{z}) \\ \sin(\boldsymbol{\omega}_S \bullet \mathbf{z}) + \sin(\boldsymbol{\omega}_S' \bullet \mathbf{z}) \end{bmatrix}, \tag{6}$$

with parameters $\boldsymbol{\omega}_1, \boldsymbol{\omega}_1', \dots \boldsymbol{\omega}_S, \boldsymbol{\omega}_S'$ that are directly optimizable as in [18]. Noted that S is a pre-specified integer that refers to the number of spectral basis functions. As it appears, higher S gives rise to better representational capacity of GP but longer training time and larger amounts of parameters. Therefore, in practice we select the largest possible S with regards to our affordable computing resources at hand.

3. Gaussian Process Neural Networks

This section describes our proposed Gaussian Process neural networks (GPNNs), which, taking the advantage of variational approximation, leads to a practical learning algorithm that is compatible to original training mechanism of standard DNNs. In GPNNs, we employ Gaussian process layer as activations using random feature expansion [19, 20] of generalized spectral kernels at the first hidden layer, and by doing so we obtain a new form of architecture to the standard DNN model, as illustrated on Figure 1. Details are presented in the below sections.

3.1. Weight-space view of Gaussian process as a layer

One important issue associated with DNNs and artificial neural networks in general is the manual selection of suitable model structures, for example, the form of activation functions. Due to lack of automatic model selection techniques, the choice of activation functions has been largely empirically based. To address this issue, we propose to relax the restriction of using deterministic activation function by imposing randomness on functions. Drawing inspiration from deep Gaussian process (DGP) [12], in this work, we employ Gaussian process (GP) to control the distribution of non-deterministic activation functions:

$$\sigma(\mathbf{z}) \sim \mathcal{GP}(0, \phi(\mathbf{z})^T \phi(\mathbf{z}')), \tag{7}$$

where $\mathbf{z}$ and $\mathbf{z}'$ are arbitrary inputs to the activation functions, and $\phi(\cdot)$ is defined in Eq. 6 for a generalized form of kernel functions. From a weight-space view of Gaussian process (see

Algorithm 1: One stochastic forward pass of GPNN

Input: $\mathbf{x}$, $(D_1, \ldots, D_H)$, S, $\{\mathbf{W}_2, \mathbf{b}_2, \ldots, \mathbf{W}_H, \mathbf{b}_H, \mathbf{W}_{\text{out}}, \mathbf{b}_{\text{out}}\}$, $\boldsymbol{\theta}$, $\mathbf{p}$, $\mathbf{p}'$

Output: $f_{\text{DNN}}(\mathbf{x})$

for $i \leftarrow 1$ ***to*** $2S$ **do**
  for $j \leftarrow 1$ ***to*** D_{in} **do**
    sample $\boldsymbol{\epsilon}_{ij}$ from $\mathcal{N}(0,1)$
    $\hat{\mathbf{Z}}_{ij} \leftarrow \mathbf{m}_{ij} + \mathbf{s}_{ij}\boldsymbol{\epsilon}_{ij}$
  end
end
$\mathbf{Z}, \mathbf{Z}' \leftarrow$ vertically split $\hat{\mathbf{Z}}$ in half
$f_1(\mathbf{x}) \leftarrow \psi(\mathbf{Z}\mathbf{x} + \mathbf{p}, \mathbf{Z}'\mathbf{x} + \mathbf{p}')$
for $h \leftarrow 2$ ***to*** H **do**
  $f_h(\mathbf{x}) \leftarrow \mathbf{W}_h f_{h-1}(\mathbf{x}) + \mathbf{b}_h$
end
$f_{\text{DNN}}(\mathbf{x}) \leftarrow \frac{1}{\sqrt{D_H}}\mathbf{W}_{\text{out}} f_H(\mathbf{x}) + \mathbf{b}_{\text{out}}$

Chapter 2 of [15]), Eq. 7 can be rewritten as $\sigma(\mathbf{z}) = \boldsymbol{\alpha} \bullet \phi(\mathbf{z})$, where $\boldsymbol{\alpha} \sim \mathcal{N}(0, \mathbf{I}_{2S})$ with $\mathbf{I}_{2S}$ being the identity matrix with $2S$ non-zero entries. Noted that the random vector $\boldsymbol{\alpha}$ represents amplitudes of different spectral basis functions. In theory it is possible to use GP-based activation functions for all DNN hidden layers, but the inference of stacked GPs is dramatically much more expensive than inferring only one layer of GP such that advanced approximation techniques are prescribed as discussed in DGP [12]. Hence, in our initial work presented in this paper, we use a GP layer to replace the standard hidden layer. In particular, the GP layer is placed on the first layer to simplify the implementation and also to avoid the pathology issue reported in [21]. Mathematically, now the input to GP is nothing but the affine transformed input as defined in Eq. 2, giving

$$\sigma(\mathbf{z}) = \boldsymbol{\alpha} \bullet \phi(\mathbf{z}) = \boldsymbol{\alpha} \bullet \phi(\mathbf{W}_{\text{in}}\mathbf{x} + \mathbf{b}_{\text{in}}), \tag{8}$$

where $\mathbf{W}_{\text{in}} \triangleq \mathbf{W}_1$ and $\mathbf{b}_{\text{in}} \triangleq \mathbf{b}_1$. Since $\phi(\cdot)$ involves a dot product to optimizable parameters $\{\boldsymbol{\omega}_1, \boldsymbol{\omega}'_1, \ldots \boldsymbol{\omega}_S, \boldsymbol{\omega}'_S\}$, we combine $\{\boldsymbol{\omega}_i, \boldsymbol{\omega}'_i\}$ with the matrix parameter $\mathbf{W}_{\text{in}}$ to obtain

$$\mathbf{Z} = \begin{bmatrix} \boldsymbol{\omega}_1^\top \\ \vdots \\ \boldsymbol{\omega}_S^\top \end{bmatrix} \mathbf{W}_{\text{in}}, \ \mathbf{Z}' = \begin{bmatrix} \boldsymbol{\omega}'^\top_1 \\ \vdots \\ \boldsymbol{\omega}'^\top_S \end{bmatrix} \mathbf{W}_{\text{in}} \in \mathbb{R}^{S \times D_{\text{in}}}, \tag{9}$$

and also with the vector parameter $\mathbf{b}_{\text{in}}$ to obtain $\mathbf{p}$ and $\mathbf{p}'$. Now, we simplify Eq. 8 into $\phi(\mathbf{z}) = \psi(\mathbf{Z}\mathbf{x} + \mathbf{p}, \mathbf{Z}'\mathbf{x} + \mathbf{p}')$ where

$$\psi(\mathbf{z}, \mathbf{z}') = \frac{1}{\sqrt{S}} \begin{bmatrix} \cos(\mathbf{z}) + \cos(\mathbf{z}') \\ \sin(\mathbf{z}) + \sin(\mathbf{z}') \end{bmatrix}. \tag{10}$$

Since in the next layer $\sigma(\mathbf{z})$ will be affine transformed given $\mathbf{W}_2$ and $\mathbf{b}_2$, optimizing the amplitudes $\boldsymbol{\alpha}$ is similar to optimizing the $\mathbf{W}_2$ in standard DNNs, and thus has less effects on non-deterministic activation functions. To maintain the randomness of activation functions, we model the distribution of spectral frequencies $p(\hat{\mathbf{Z}}|\mathbf{X}, \mathbf{Y})$ with $\hat{\mathbf{Z}} = \begin{bmatrix} \mathbf{Z} \\ \mathbf{Z}' \end{bmatrix}$ instead of modeling the distribution of amplitudes $\boldsymbol{\alpha}$.

3.2. Training of GPNN in standard DNN framework

We embrace variantional inference using a variational distribution of $\hat{\mathbf{Z}}$:

$$q(\hat{\mathbf{Z}}_{ij}) = \mathcal{N}\left(\mathbf{m}_{ij}, \mathbf{s}_{ij}^2\right) \tag{11}$$

with variational parameters. For ease of reference, we use $\boldsymbol{\theta}$ denote the collection of all $\boldsymbol{\theta}_{ij}$ for all $i \in \{1, \ldots, 2S\}, j \in \{1, \ldots, D_{\text{in}}\}$. To perform inference over GPNN, our task is to find the variational parameters to represent the original distribution $p(\hat{\mathbf{Z}}|\mathbf{X}, \mathbf{Y})$. In variantional inference [22], $\boldsymbol{\theta}$ can be directly learned from back-propagation by differentiating the negative evidence lower bound as objective function of minimization:

$$\mathcal{L}(\boldsymbol{\theta}) = -\mathbb{E}_{q(\hat{\mathbf{Z}};\boldsymbol{\theta})}\left[\log p(\mathbf{Y}|\mathbf{X}, \hat{\mathbf{Z}})\right] + \mathcal{KL}\left(q(\hat{\mathbf{Z}};\boldsymbol{\theta})||p(\hat{\mathbf{Z}})\right), \tag{12}$$

where $\mathcal{KL}(\cdot||\cdot)$ is the Kullback-Leibler (KL) divergence that measures the distance between distributions. We refer to the setting in [20, 23] where the prior $p(\hat{\mathbf{Z}})$ as *i.i.d* Gaussian distribution, i.e., $p(\hat{\mathbf{Z}}_{ij}) \sim \mathcal{N}(0,1)$. Using stochastic gradient variantional Bayes estimator [24], we apply a re-parameterization trick for each entry of the random matrix $\hat{\mathbf{Z}}$ using a differentiable function $g(\cdot)$:

$$\hat{\mathbf{Z}}_{ij} = g(\boldsymbol{\epsilon}_{ij}) = \mathbf{m}_{ij} + \mathbf{s}_{ij}\boldsymbol{\epsilon}_{ij}, \ \boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(0,1). \tag{13}$$

Then, we have

$$\mathbb{E}_{q(\hat{\mathbf{Z}};\boldsymbol{\theta})}\left[\log p(\mathbf{Y}|\mathbf{X}, \hat{\mathbf{Z}})\right] = \mathbb{E}_{\boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(0,1)}\left[\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\epsilon})\right] \tag{14}$$

meaning that the source of randomness comes from $\boldsymbol{\epsilon}_{ij}$, which can be easily generated in modern programming libraries. Here, the definition of $\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\epsilon})$ depends on which the task GPNN is tackling. In our application to speech recognition tasks where Y is the tied triphone states, we define:

$$\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\epsilon}) = \sum_{i=1}^{N} \mathbf{y}_i \bullet \log \text{softmax}\left(f_{\text{DNN}}(\mathbf{x}_i)\right), \tag{15}$$

where $(\mathbf{x}_i, \mathbf{y}_i)$ is the ith training input-output pair. It is essential that above expectation term resembles the typical cross-entropy used in the standard DNN framework. What's more, KL divergence can be regarded as a regularization term for the GP layer [25], which again falls into the original DNN framework. These together implies that, we can train GPNN in exactly the same DNN framework, where a number of programming tools are available for. All in all, one stochastic forward pass of GPNN is shown on Algorithm 1. At training time, we can do a forward pass and followed by the standard back-propagation algorithm to update the parameters as in standard DNNs. But, at testing time, we need to run the forward pass algorithm T times and average all T predictions so as to marginalize over all possible activations.

4. Performance Analysis

This section aims to compare our proposed GPNN acoustic model to DNN acoustic model on the ARPA Resource Management (RM) speech dataset. The number of nodes is varied from 50 to 4000 to demonstrate the advantage of GPNNs in modeling structural and parametric uncertainty to prevent over-fitting. We analyzed GPNN's and DNN's ability of capturing distinct features characterizing different speech patterns. The effects of the number of spectral basis functions on the performance of GPNN systems is also investigated.

Table 1: *WER (%) of four test sets on RM in DNN and GPNN system by varying the number of hidden nodes (D_h) and the number of spectral basis functions (S). N denotes the number of free parameters in first hidden layer.*

		DNN						GPNN						Improve
S	D_h	N	feb89	feb91	sep92	oct89	Total	N	feb89	feb91	sep92	oct89	Total	Total
25	50	17600	11.79	10.39	17.27	12.30	12.95	17600	**7.81**	**8.33**	**13.21**	**9.84**	**9.81**	+24.25%
60	125	44000	7.97	8.53	14.38	10.47	10.35	44000	**7.54**	**7.93**	**13.48**	**9.31**	**9.57**	+7.54%
125	250	88000	7.18	7.13	12.66	8.98	9.00	88000	**6.33**	**6.80**	**12.31**	**8.94**	**8.61**	+4.33%
250	500	176000	5.97	**6.32**	11.61	**7.68**	7.90	176000	**5.58**	6.44	**10.39**	7.71	**7.54**	+4.56%
500	1000	352000	**5.51**	6.40	10.28	8.27	7.63	352000	5.58	**5.88**	**9.69**	**7.53**	**7.18**	+5.90%
750	1500	528000	5.62	6.32	10.43	7.90	7.58	528000	5.62	**5.64**	**9.42**	**7.34**	**7.02**	+7.39%
1000	2000	704000	5.47	**6.08**	9.57	8.23	7.36	704000	**5.23**	6.56	**9.30**	**7.30**	**7.11**	+3.40%
2000	4000	1408000	6.56	6.88	10.86	8.35	8.17	1408000	**5.08**	**6.20**	**8.36**	**6.86**	**6.63**	+18.85%

Table 2: *WER (%) performance of four test sets on RM in GPNN system with different number of spectral basis functions (S)*

S	feb89	feb91	sep92	oct89	Total
125	6.17	6.64	10.47	7.75	**7.77**
250	5.58	6.44	10.39	7.71	**7.54**
1000	5.54	6.16	9.34	7.04	**7.03**

4.1. Datasets and experimental setup

We conducted experiments on RM dataset. The input features consist of 13 Mel-frequency cepstral coefficients (MFCC) concatenated with their first and second temporal differentials by making use of a context of 9 frames, this leads to the construction of a 351 dimensional acoustic input feature at each frame. The output consists of 689 tied tri-phone states.

Hybrid DNN baseline systems were created using the HTK [26] open-source toolkit in two steps. At first step, we did layer-wise discriminative pretraining with learning rate 0.001 using the LIST learning rate scheduler, and then fin-tuned network free parameters with learning rate 0.002 based on NEWBOB learning rate scheduler. On the other hand, GPNN was implemented in PyTorch [27]. We replaced the first hidden layer of standard hybrid DNN by a GP activation layer, and the rest of the model architecture remained the same as DNN baseline system. We used Adam [28] optimization algorithm with learning rate 0.001 and early stopping strategy . In training of both networks, cross-entropy was chosen as the loss function. In decoding, a tri-gram language model constructed using all RM acoustic data transcripts was used for both DNN and GPNN systems.

4.2. Varying the number of hidden nodes

We trained DNNs and GPNNs with 5 hidden layers, and kept the number of hidden units same across each hidden layer. In GPNN, the first hidden layer is GP layer while the other layers remain the same as baseline DNNs. Varying the number of nodes in each hidden layer corresponds to different structures. Since GPNNs use more generalized activation functions, it could perform more robustly than DNN. In order to show its robustness to underfitting and overfitting, we vary the number of hidden nodes D_h in all layers ($h = 1, \ldots, H$) from 50 to 4000. Table 1 shows word error rate (WER) performance of DNN and GPNN systems with 50, 125, 500, 1000, 1500, 2000 and 4000 nodes in each hidden layer. As expected, DNNs with fewer hidden nodes performed worse due to its lack of capacity of understanding complex relationship encoded in the speech data. On the other hand, when the number of hidden nodes is too large, there is higher risk of overfitting in DNN system.

From the results, consistent performance improvements were achieved by GPNN system over the baseline DNNs. Especially when the number of hidden units equals 50 and 4000, where GPNN system relatively improved 24.25% and 18.85% of the WER over DNN baseline system, respectively. This is in line with our expectation – DNNs with only a few hidden nodes cannot capture the complex mapping between acoustic features to triphone states, but GPNNs are able to mitigate under-fitting when under a highly punitive constraint of model complexity. Also, we see that DNN tends to be over-trained when too many hidden nodes are used, while GPNN still retains a good performance. This clearly demonstrates the ability of GPNNs to prevent overfitting when building large-sized networks. Based on the experiment results, we note that GPNNs can capture more complex relationship between acoustic input and target triphone state labels for using more generalized activation function forms. Considering these, we conclude that GPNNs are more robust than DNNs.

4.3. Varying the number of spectral basis functions

In GP activation functions, the number of spectral basis functions plays a vital role in representing possible forms of activation functions. Nonetheless, we varied the number of spectral basis functions in the GP layer while fixing other hidden layers. Table 2 shows the WER performance of GPNN systems containing 125, 250, 1000 spectral basis functions in RM dataset, while fixing 500 hidden nodes in all hidden layers. From the result, in all four test sets, WERs were decreased with larger number of spectral basis functions. This implies that the GP layer gained more generalization power because of possessing more spectral basis functions. With such an effect, it is encouraging to use larger number of spectral basis functions in GPNNs.

5. Conclusions

In this paper, we propose a novel structure of neural network called Gaussian process neural networks (GPNN) for speech recognition using non-parametric activation functions based on GP. Benefiting from the GP layer, the problem of model structural and parametric uncertainty are addressed. We varied the number of hidden nodes, and obtained consistent improvements of GPNN over the DNN baseline. From these experiments, we conclude that GPNNs are more robust than standard DNNs. We found that DNNs with fewer or larger number of hidden nodes are prone to underfitting or overfitting, while GPNNs can still give high standard of performances. Moreover, we analyzed the effects of the number of spectral basis functions S on GPNN's performance. The results suggest that we can boost the performance of GPNN by continuously increasing S. In the future, we are eager to apply GPNNs to large vocabulary recognition tasks such as Switchboard. It is also worthy to investigate the recurrent formulation of GPNN for full sequence learning.

6. References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] K. J. Han, S. Hahm, B.-H. Kim, J. Kim, and I. Lane, "Deep learning-based telephony speech recognition in the wild," in *Proc. Interspeech*, 2017, pp. 1323–1327.

[3] G. Saon, G. Kurata, T. Sercu, K. Audhkhasi, S. Thomas, D. Dimitriadis, X. Cui, B. Ramabhadran, M. Picheny, L.-L. Lim *et al.*, "English conversational telephone speech recognition by humans and machines," *arXiv preprint arXiv:1703.02136*, 2017.

[4] A. H. Abdelaziz, S. Watanabe, J. R. Hershey, E. Vincent, and D. Kolossa, "Uncertainty propagation through deep neural networks," in *Interspeech 2015*, 2015.

[5] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[6] R. M. Neal, "Priors for infinite networks," in *Bayesian Learning for Neural Networks*. Springer, 1996, pp. 29–53.

[7] T. Hazan and T. Jaakkola, "Steps toward deep kernel methods from infinite neural networks," *arXiv preprint arXiv:1508.05133*, 2015.

[8] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," *arXiv preprint arXiv:1711.00165*, 2017.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] G. Pundak and T. N. Sainath, "Highway-lstm and recurrent highway networks for speech recognition," in *Proceedings of Interspeech*, 2017.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[12] A. Damianou and N. Lawrence, "Deep gaussian processes," in *Artificial Intelligence and Statistics*, 2013, pp. 207–215.

[13] Y.-L. K. Samo and S. Roberts, "Generalized spectral kernels," *arXiv preprint arXiv:1506.02236*, 2015.

[14] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[15] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[16] D. Duvenaud, "Automatic model construction with gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.

[17] Y.-L. Kom Samo and S. Roberts, "Generalized spectral kernels," *arXiv preprint arXiv:1506.02236*, 2015.

[18] J. Làzaro-Gredilla, C. E. Rasmussen, A. R. Figueiras-Vidal *et al.*, "Sparse spectrum gaussian process regression," *Journal of Machine Learning Research*, vol. 11, no. Jun, pp. 1865–1881, 2010.

[19] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[20] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone, "Random feature expansions for deep gaussian processes," *arXiv preprint arXiv:1610.04386*, 2016.

[21] D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani, "Avoiding pathologies in very deep networks," in *Artificial Intelligence and Statistics*, 2014, pp. 202–210.

[22] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[23] Y. Gal, R. T. McAllister, and C. E. Rasmussen, "Improving pilco with bayesian neural network dynamics models," in *Data-Efficient Machine Learning workshop*, vol. 951, 2016, p. 2016.

[24] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[25] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7893–7897.

[26] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey *et al.*, "The htk book," *Cambridge university engineering department*, vol. 3, p. 175, 2002.

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.