BAYESIAN AND GAUSSIAN PROCESS NEURAL NETWORKS FOR LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

Shoukang Hu, Max W. Y. Lam, Xurong Xie, Shansong Liu, Jianwei Yu, Xixin Wu, Xunying Liu, Helen Meng

The Chinese University of Hong Kong, Hong Kong SAR, China

{skhu, wylam, ssliu, jwyu, xxwu, xyliu, hmmeng}@se.cuhk.edu.hk {xrxie}@ee.cuhk.edu.hk

ABSTRACT

The hidden activation functions inside deep neural networks (DNNs) play a vital role in learning high level discriminative features and controlling the information flows to track longer history. However, the fixed model parameters used in standard DNNs can lead to over-fitting and poor generalization when given limited training data. Furthermore, the precise forms of activations used in DNNs are often manually set at a global level for all hidden nodes, thus lacking an automatic selection method. In order to address these issues, Bayesian neural networks (BNNs) acoustic models are proposed in this paper to explicitly model the uncertainty associated with DNN parameters. Gaussian Process (GP) activations based DNN and LSTM acoustic models are also used in this paper to allow the optimal forms of hidden activations to be stochastically learned for individual hidden nodes. An efficient variational inference based training algorithm is derived for BNN, GPNN and GPLSTM systems. Experiments were conducted on a LVCSR system trained on a 75 hour subset of Switchboard I data. The best BNN and GPNN systems outperformed both the baseline DNN systems constructed using fixed form activations and their combination via frame level joint decoding by **1%** absolute in word error rate.

Index Terms— Bayesian Neural Network, Gaussian Process Neural Network, activation function selection, speech recognition

1. INTRODUCTION

There has been a long history of interest in applying artificial neural networks (ANNs) in speech recognition systems [1, 2, 3, 4]. In recent years, the rapid advanced deep learning technologies [5] allow them to be widely used in state-of-the-art speech recognition systems [6, 7, 8, 9].

Two issues arise when designing deep neural network (DNN) based speech recognition systems. First, fixed parameter based DNNs can lead to over-fitting and poor generalization when given limited data. Second, there have been a wide range of activation functions developed over years to improve the performance of DNN based recognition systems. The early forms of hybrid DNN systems used Sigmoid activation functions. Rectified Linear Units (ReLUs) [10, 11] were proposed in recent years to achieve fast convergence in training. In order to model long-range temporal dependencies in speech data, long short-term memory (LSTM) cells [12, 13] were also widely used in recurrent neural network based speech recognition systems.

In order to address the issue associated with parameter uncertainty, Bayesian Neural Networks (BNNs) were commonly adopted. In the machine learning community, a series of previous research were conducted in this direction. A practical Bayesian framework for back-propagation networks was introduced in [14]. Efficient variational learning based inference was later proposed in [15] for BNNs. In contrast, limited research has been conducted to apply BNNs for speech recognition systems. In [16], a Bayesian recurrent neural network (RNN) using variational inference based training was evaluated on the TIMIT corpus. A Bayesian learning approach for RNN language models was also proposed in [17].

In terms of the uncertainty associated with activation functions, previous research is more limited. A fixed, deterministic weighting based multiple hidden activations combination method was proposed in [18]. A special case of Gaussian Process Neural Networks (GPNNs) using a fixed weight combination of multiple Bayesian Neural Networks [19] was proposed in our earlier research. However, in both cases, the combination weights for activation selection were fixed-point estimates, thus considering no parameter uncertainty given limited data. In order to address this issue, in this paper we propose more general forms of GPNNs to handle the uncertainty associated with both activation weight parameters and their combination weights. Gaussian Process based LSTM acoustic models are also proposed.

To the best of our knowledge, this paper is the first attempt to apply BNN, GPNN and GPLSTM acoustic models for LVCSR tasks. Our previous research of GPNN reported in [19] was a simplified case of the more general forms of GPNN systems considered in this paper, and only evaluated on a smaller vocabulary continuous speech recognition task on the Resource Management corpus.

The rest of this paper is organized as follows. Section 2 and Section 3 introduce the DNN and BNN models. GPNN and GPLSTM models are proposed in Section 4 and Section 5. A variational inference based efficient training algorithm is presented in Section 6. Section 7 presents the experiments and results. Finally, the conclusions are drawn in Section 8.

2. DEEP NEURAL NETWORK

Deep Neural Networks (DNNs) learn the fixed-point parameter estimates through the training data $\{\mathbf{x}_t, \widehat{\mathbf{y}}_t\}$. Given an input vector $\mathbf{z}^{(l-1)}$ from $(l-1)$-th layer, a standard DNN in Table 1 (second line) computes the output $h_i^{(l)}(\mathbf{z}^{(l-1)})$ of the i-th node in the l-th layer by the following equation.

$$h_i^{(l)}(\mathbf{z}^{(l-1)}) = \phi\left(\mathbf{w}_i^{(l)} \bullet \mathbf{z}^{(l-1)}\right) \tag{1}$$

where, $\mathbf{z}^{(l-1)} = \left[h_1^{(l-1)}(\mathbf{z}^{(l-2)}), \cdots, h_d^{(l-1)}(\mathbf{z}^{(l-2)}), 1\right]^T$ is the input vector fed into the l-th hidden layer, $\mathbf{w}_i^{(l)} = \left[w_{i,1}^{(l)}, \cdots, w_{i,d}^{(l)}, b\right]^T$

This research was partially funded by Research Grants Council of Hong Kong General Research grant Fund No.14200218, and the Chinese University of Hong Kong (CUHK) grant No. 4055065.

System	Uncertainty		#Parameter
	$\boldsymbol{\lambda}$	$\mathbf{w}$	
DNN	$\times$	$\times$	ab
DNN-JointDec[20, 21]	$\times$	$\times$	$3ab$
BNN	$\times$	$\surd$	$ab+a$
GPNN-0	$\times$	$\times$	$ab+3b$
GPNN-1	$\surd$	$\times$	$ab+3+3b$
GPNN-2[19]	$\times$	$\surd$	$ab+a+3b$
GPNN-3	$\surd$	$\surd$	$ab+a+3b+3$
LSTM	$\times$	$\times$	$4ab+4b^2+4b$
GPLSTM-1	$\surd$	$\times$	$4ab+4b^2+24(b+1)$

Table 1. The forms of parameter uncertainty considered and the number of free parameters w.r.t. the number of hidden nodes per layer in different DNN, BNN and GPNN systems, assuming the input vector size is a and the number of nodes is b.

denotes the node's weight vector, $\phi(\cdot)$ is the activation function, and $\bullet$ denotes the dot product.

In this paper, three widely used activation functions are employed as our basis activation functions, i.e., Sigmoid, Tanh, ReLU. A simple form of stochastic activation function selection uses a linear interpolation over all the basis activation functions as

$$h_i^{(l)}(\mathbf{z}^{(l-1)}) = \sum_m \lambda_i^{(l,m)} \phi_m\left(\mathbf{w}_i^{(l,m)} \bullet \mathbf{z}^{(l-1)}\right) \tag{2}$$

where $\lambda_i^{(l,m)}$ is the m-th basis activation coefficient, $\mathbf{w}_i^{(l,m)}$ is the fixed weight parameter and $\phi_m(\cdot)$ is the m-th basis activation function. When given limited data, directly learning of weight parameters $\left\{\mathbf{w}_i^{(l,m)}\right\}$ can lead to over-fitting and poor generalization.

3. BAYESIAN NEURAL NETWORK

In order to address the over-fitting and poor generalization problem above, a Bayesian Neural Network (BNN) can be used. Instead of using fixed-point estimates of weight parameters, posterior distributions are used to model the uncertainty associated with weight parameters [22]. The expected hidden node output is marginalized over different parameter estimates.

$$h_i^{(l)}(\mathbf{z}^{(l-1)}) = \int \phi\left(\mathbf{w}_i^{(l)} \bullet \mathbf{z}^{(l-1)}\right) p(\mathbf{w}_i^{(l)}) d\mathbf{w}_i^{(l)} \tag{3}$$

where $p(\mathbf{w}_i^{(l)}) = p(\mathbf{w}_i^{(l)} \mid \{\mathbf{x}_t, \widehat{\mathbf{y}}_t\})$ denotes the node dependent activation parameter posterior distribution to be learned from training data, $\phi(\cdot)$ is the activation function. One key issue associated with BNN is the selection of activation functions. This is normally deterministic and requires expert knowledge.

4. GAUSSIAN PROCESS NEURAL NETWORK

Not only the weight parameters inside activation functions can be regarded as uncertain variables in BNNs, we can also regard the basis coefficients as additional uncertain variables to be integrated over. Thus the deterministic combination in Eqn.(2) is modified into double integration of both weight and coefficient variables in Eqn.(4).

$$\begin{aligned} h_i^{(l)}(\mathbf{z}^{(l-1)}) = \sum_m \int\int \lambda_i^{(l,m)} \phi_m\left(\mathbf{w}_i^{(l,m)} \bullet \mathbf{z}^{(l-1)}\right) \\ p(\mathbf{w}_i^{(l,m)}) p(\lambda_i^{(l,m)}) d\mathbf{w}_i^{(l,m)} d\lambda_i^{(l,m)} \end{aligned} \tag{4}$$

where $p(\lambda_i^{(l,m)}) = p(\lambda_i^{(l,m)} \mid \{\mathbf{x}_t, \widehat{\mathbf{y}}_t\})$ and $p(\mathbf{w}_i^{(l,m)}) = p(\mathbf{w}_i^{(l,m)} \mid \{\mathbf{x}_t, \widehat{\mathbf{y}}_t\})$ denote the basis activation coefficient and parameter posterior distributions respectively, and we assume the statistical independence between these two variables. The general form of Gaussian Process neural networks (GPNNs[1]) in Eqn.(4) proposed in this paper subsuming the previous work in [19] can be simplified to three different special cases depending on the parameter uncertainty been considered.

4.1. Both $\boldsymbol{\lambda}$ and $\mathbf{w}$ are deterministic

In this case, both the basis activation coefficients $\left\{\lambda_i^{(l,m)}\right\}$ and weight parameters $\left\{\mathbf{w}_i^{(l,m)}\right\}$ are deterministic. This is the fixed weight based system shown in Eqn.(2), also shown as GPNN-0 system (fifth line) in Table 1.

4.2. Treating $\boldsymbol{\lambda}$ as uncertain

When only $\{\lambda_i^{(l,m)}\}$ are treated as random variables, Eqn.(4) can be simplified to the integration over basis coefficients in the following form. This is shown as the GPNN-1 system (sixth line) in table 1.

$$h_i^{(l)}(\mathbf{z}^{(l-1)}) = \sum_m \int \lambda_i^{(l,m)} \phi_m\left(\mathbf{w}_i^{(l,m)} \bullet \mathbf{z}^{(l-1)}\right) p(\lambda_i^{(l,m)}) d\lambda_i^{(l,m)} \tag{5}$$

where $\mathbf{w}_i^{(l,m)}$ takes a fixed-point estimate.

4.3. Treating $\mathbf{w}$ as uncertain

This is the precursor form of Gaussian Process Neural Network we propose in the early research [19]. This simplified form can be interpreted by a standard weight-space view of Gaussian Process [23].

$$h_i^{(l)}(\mathbf{z}^{(l-1)}) = \sum_m \lambda_i^{(l,m)} \int \phi_m\left(\mathbf{w}_i^{(l,m)} \bullet \mathbf{z}^{(l-1)}\right) p(\mathbf{w}_i^{(l,m)}) d\mathbf{w}_i^{(l,m)} \tag{6}$$

where $\lambda_i^{(l,m)}$ takes a fixed-point estimate.

5. GAUSSIAN PROCESS LSTM-RNN

Using gated functions in cells, LSTM controls the information flow and tracks longer context information that are useful to predict the following speech frames. However, the precise form of activations inside the LSTM has been set for all cells at a global level. Inspired by the Gaussian Process (GP) based recurrent neural network investigated in language modeling [24], we use GP based activations to replace the standard linear activation weights multiplied with the hidden vector input at a previous time step in LSTM cells, which is illustrated in Figure 1. A GPLSTM network computes a mapping from the input sequence of vectors $\{x_1, \cdots, x_T\}$ to an output sequence $\{h_1, \cdots, h_T\}$ by calculating the network unit activation using the following equations iteratively from $t = 1$ to T:

$$\begin{aligned} \mathbf{i}_t &= \boldsymbol{\sigma}(\mathbf{W}_{ix}\mathbf{x}_t + \boldsymbol{\Theta}_{ih}^{gp}(\mathbf{h}_{t-1})) \\ \mathbf{f}_t &= \boldsymbol{\sigma}(\mathbf{W}_{fx}\mathbf{x}_t + \boldsymbol{\Theta}_{fh}^{gp}(\mathbf{h}_{t-1})) \\ \widetilde{\mathbf{c}}_t &= \boldsymbol{\sigma}(\mathbf{W}_{\widetilde{c}x}\mathbf{x}_t + \boldsymbol{\Theta}_{\widetilde{c}h}^{gp}(\mathbf{h}_{t-1})) \\ \mathbf{o}_t &= \boldsymbol{\sigma}(\mathbf{W}_{ox}\mathbf{x}_t + \boldsymbol{\Theta}_{oh}^{gp}(\mathbf{h}_{t-1})) \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{c}}_t \\ \mathbf{h}_t &= \mathbf{o}_t \odot \mathbf{tanh}(\mathbf{c}_t) \end{aligned} \tag{7}$$

[1]The GPNN can be seen to combine multiple basis BNN activation outputs.

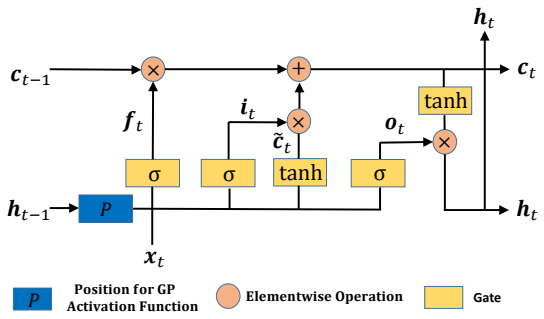

Fig. 1. An example of the architecture of a GPLSTM cell (GP activation position highlighted as blue)

where the $\mathbf{W}_{ix}$, $\mathbf{W}_{fx}$, $\mathbf{W}_{\tilde{c}x}$, $\mathbf{W}_{ox}$ are the matrix weights from input gate, forget gate, cell activation output vector and output gate to input, $\mathbf{i}, \mathbf{f}, \mathbf{o}, \mathbf{c}, \mathbf{h}$ are respectively the input gate, forget gate, output gate, cell state vector and hidden state vector, $\boldsymbol{\Theta}^{gp}$ is the GPNN output, $\odot$ is the element-wise multiplication. As is shown in Figure 1 in the left bottom corner, we apply a GPNN layer to map the hidden state $\mathbf{h}_{t-1}$ to a same dimensional space, e.g., $\boldsymbol{\Theta}^{gp}_{ih}(\mathbf{h}_{t-1})$ is the output of one GPNN layer by feeding $\mathbf{h}_{t-1}$ as the input vector.

6. VARIATIONAL INFERENCE BASED EFFICIENT TRAINING FOR BNN AND GPNN SYSTEMS

The general forms of BNN in Eqn.(3), GPNN in Eqn.(4) and GPLSTM in Eqn.(7) require integration over random variables regarded as uncertain. Since the distributions of these random variables are intractable, there is no such closed form of the integral. Due to its non-differentiability, direct application of back-propagation algorithm is not applicable. Several techniques can be adopted to approximate this integration, for example, the Laplace approximation using Gaussian shape proposed in [17]. In this paper, we apply the variational inference approach [25] to solve this issue. And we assume that the variational distribution and the prior distribution are both Gaussian distributions following the work in [15]. To introduce the variational inference approach, we take BNN model for an example. For notation simplicity, we consider the parameters $\mathbf{w} = \mathbf{w}_i^{(l,m)}$ as the m-th basis of the i-th node in the layer l.

Applying Jensen's inequality, we can calculate the evidence lower bound of the conditional log-likelihood:

$$\begin{aligned}\log P(\mathbf{y} \mid \mathbf{x}) &= \log \int P(\mathbf{y}\,|\,\mathbf{w},\mathbf{x}) P_r(\mathbf{w}) d\mathbf{w} \\ &\geq \underbrace{\int q(\mathbf{w}) \log P(\mathbf{y}\,|\,\mathbf{w},\mathbf{x}) d\mathbf{w}}_{\mathcal{L}_1} - \underbrace{KL(q(\mathbf{w}) \| P_r(\mathbf{w}))}_{\mathcal{L}_2} = \mathcal{L}\end{aligned} \tag{8}$$

where $P_r(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\sigma}_r^2)$ denotes the weight prior distribution, $q(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ is the variational approximation of the parameter posterior distribution $p(\mathbf{w})$, $KL(q\|P_r)$ is the Kullback-Leibler (KL) divergence between q and P_r.

The first term $\mathcal{L}_1$ in Eqn.(8) can be efficiently approximated by Monte Carlo (MC) sampling method.

$$\mathcal{L}_1 \approx \frac{1}{N} \sum_{k=1}^{N} \log P(\mathbf{y}\,|\,\boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_k, \mathbf{x}) \tag{9}$$

where $\boldsymbol{\epsilon}_k = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the k-th sample.

The KL divergence between q and P_r of the second term $\mathcal{L}_2$ in Eqn.(8) can be simplified as follows,

$$\mathcal{L}_2 = \sum_j \left\{ \log \frac{\sigma_{r,j}}{\sigma_j} + \frac{\sigma_j^2 + (\mu_j - \mu_{r,j})^2}{2\sigma_{r,j}^2} - \frac{1}{2} \right\} \tag{10}$$

where μ_j and σ_j are the j-th component of variational posterior distribution hyper-parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\mu_{r,j}$ and $\sigma_{r,j}$ are the j-th component of prior distribution hyper-parameters $\boldsymbol{\mu}_r$ and $\boldsymbol{\sigma}_r$.

The gradient statistics can be computed for the hyper-parameters $\theta = \{\mu_j, \sigma_j\}$ as below.

$$\frac{\partial \mathcal{L}}{\partial \mu_j} = \frac{1}{N} \sum_{k=1}^{N} \frac{\partial \log P(\mathbf{y}\,|\,\mathbf{x}, \theta, \boldsymbol{\epsilon}_k)}{\partial \mu_j} - \frac{\mu_j - \mu_{r,j}}{\sigma_j^2} \tag{11}$$

$$\frac{\partial \mathcal{L}}{\partial \sigma_j} = \frac{1}{N} \sum_{k=1}^{N} \frac{\partial \log P(\mathbf{y}\,|\,\mathbf{x}, \theta, \boldsymbol{\epsilon}_k)}{\partial \sigma_j} - \frac{\sigma_j^2 - \sigma_{r,j}^2}{\sigma_j \sigma_{r,j}^2} \tag{12}$$

where $\frac{\partial \log P(\mathbf{y}|\mathbf{x},\theta,\boldsymbol{\epsilon}_k)}{\partial \mu_j}$, $\frac{\partial \log P(\mathbf{y}|\mathbf{x},\theta,\boldsymbol{\epsilon}_k)}{\partial \sigma_j}$ can be directly calculated using the standard back-propagation method.

An important issue when training DNN, BNN and GPLSTM systems is the parameter prior to use. The choice of prior has an impact on model convergence and training efficiency. In this paper, we set the priors for BNN systems to be based on the standard DNN systems using the same activations with fixed parameters. The priors of the GPNN systems are based on the fixed weighted activation combination of Eqn.(2) (also shown as GPNN-0 system in table 1).

When evaluating the BNN and GPNN systems, the means of the respective activation weight and basis coefficient parameter posterior distributions are taken to calculate the integrals in $\mathcal{L}_1$ in a forward pass. The frame level output probability tables are then fed to the **HDecode** in the HTK toolkit [26] to produce the recognition outputs.

7. EXPERIMENTS

This section describes our experiments carried out on the 75 hour Switchboard I data to evaluate the performance of DNN, BNN, GNNN and GPLSTM systems.

7.1. Experimental Setup

Our 75 hour Switchboard I data consisted of randomly selected 1082 conversational sides out of the 4870 speakers from the 300 hour Switchboard I corpus. On the same 300 hours data, if using all the training speakers, the word error rates (WERs) of previous published DNN and TDNN systems [27, 28] on the **swbd** subset of the hub5 2000 evaluation set were reported to be 15.1 and 14.0. By using the same 300 hour full set and a four-gram language model (LM) trained on the Switchboard and Fisher transcripts, a cross-entropy (CE) trained stacked hybrid DNN baseline system gave a WER of **13.5** on the **swbd** set. A comparable version of this stacked DNN system was then trained on the 75 hour subset and used as the baseline DNN system in our experiments.

In our experiments, all systems used 6062 decision tree clustered triphone states as the output targets. In common with the bottleneck (BN) features used in a tandem system [29], we concatenated 39 dimensional BN features learned from a DNN with 13 dimensional PLP features including differential parameters up to the third order as the stacked DNN system inputs. A 9 frame context window was used. Both the bottleneck feature DNN and the stacked DNN used 6

non-constrictive hidden layers of 2000 neurons each. During training, we performed a layer-wise discriminative pretraining, then finetuned the whole network using a NEWBOB learning rate scheduler. For performance evaluation, we used both the Switchboard (**swbd**) and CallHome (**callhm**) subsets of the HUB5 2000 evaluation set.

All our DNN, BNN, GPNN and LSTM based systems were optimized using stochastic gradient descent (SGD) with momentum in PyTorch [30]. In order to obtain a fair comparison, the BNN and GPNN systems shared the same model structure as the DNN systems except the first hidden layer was modified to use the Bayesian activation of Eqn.(3) or the GP activations of section 4. In the LSTM and GPLSTM systems, we replaced the last two hidden layers of the DNN system with a comparable LSTM or GPLSTM layer. Based on the DNN system performances of Table 2 using Sigmoid or other activations, we decided to fix all the other subsequent non-GPNN layers using Sigmoid activation functions. In order to retain a comparable number of free parameters, the hyper-parameter $\boldsymbol{\mu}$ was shared within the same hidden node and $\boldsymbol{\sigma}$ was shared among all the hidden nodes in the same layer.

7.2. Performance of BNN systems

In this section, we compare the performance of various DNN baseline systems and comparable BNN systems constructed using three basis activation functions, i.e., Sigmoid, ReLU and Tanh. These are shown in Table 2. There are two main trends observed in the results of Table 2. First, irrespective of the activation functions being used, the BNN systems consistently outperformed the DNN baseline systems with the same activation functions. Second, there was also a large difference in performance between systems using different activation functions.

System	Activation	FrAcc(%)	WER(%)	
			swbd	**callhm**
DNN	Sigmoid	58.16	18.0	32.5
	ReLU	57.51	18.1	33.3
	Tanh	48.34	18.9	33.2
BNN	Sigmoid	58.29	**17.1**	**32.3**
	ReLU	56.41	18.0	33.1
	Tanh	50.81	17.9	32.8

Table 2. Performance comparison of baseline DNN and BNN systems on the **swbd** and **callhm** data

7.3. Performance of GPNN systems

A frame level joint decoding [20, 21] based combination system that equally combined the log-posterior probabilities of Sigmoid, Tanh and ReLU DNN systems served as a baseline activation combination method shown in the second line of Table 3. The performance of four different GPNN systems presented in Section 4 is shown in the remaining part of Table 3. Several trends can be observed from the results of Table 3. First, fixed weight parameters based GPNN-0 system presented in Section 4 and Eqn.(2) produced a large improvement on the **swbd** subset by **0.9%** absolute over the DNN-JointDec system, although no improvements on the **callhm** subset. Second, modeling parameter uncertainty either using GPNN-1 system for basis activation coefficients $\{\boldsymbol{\lambda}_i^{(l)}\}$, or modeling the uncertainty over the weight parameters $\{\mathbf{w}_i^{(l)}\}$ in the GPNN-2 system, consistent improvements were further obtained over the GPNN-0 system by **0.8%** absolute on the **callhm** subset. Finally, the best performance was obtained with the GPNN-3 system which modelled the uncertainty associated with both basis coefficients and weight parameters shown in Eqn.(4). Compared with the DNN baseline system using Sigmoid activation functions (second line in Table 2) as the best activation choice, the GPNN-3 system produced a **0.8%** absolute WER reduction on the **swbd** subset and a **0.7%** absolute improvement on the **callhm** subset. This system also outperformed the joint decoding system by **1%** absolute on both test sets.

System	FrAcc(%)	WER(%)	
		swbd	**callhm**
DNN-JointDec	-	18.2	32.8
GPNN-0	58.18	17.3	32.8
GPNN-1	58.78	17.4	32.0
GPNN-2[19]	58.39	17.4	32.0
GPNN-3	58.13	**17.2**	**31.8**

Table 3. Performance comparison of DNN-JointDec and GPNN systems on the **swbd** and **callhm** data

7.4. Performance of GPLSTM-RNN systems

In this section, we made an initial investigation of GPLSTM system performance on the same 75 hour training set. Its performance contrast against the baseline LSTM system is shown in Table 4. Compared with the DNN baseline systems shown in Table 2, the LSTM system produced much better cross validation data frame level accuracy. However, this system did not not produce lower word error rates.[2] In common with the trends found in Table 3, the GPLSTM system also consistently outperformed the LSTM baseline system by **0.5%** and **0.4%** absolute on both test sets.

System	FrAcc (%)	WER(%)	
		swbd	**callhm**
LSTM	62.32	18.6	34.1
GPLSTM-1	61.84	**18.1**	**33.7**

Table 4. Performance comparison of baseline LSTM and GPLSTM systems on the **swbd** and **callhm** data

8. CONCLUSIONS

In this paper, we investigated BNN, GPNN and GPLSTM acoustic models for LVCSR systems. Consistent performance improvements by up to **1%** were obtained over DNN and LSTM baseline systems with deterministic activation function and fixed parameter estimates. To the best of our knowledge, this is the first work to apply GPNN, GPLSTM acoustic models to LVCSR tasks. Future research will focus on further investigation of GPLSTM system performance and its application to other data sets.

[2] A further investigation was also made to train an alternative LSTM system with non-tandem, standard filter bank features as its inputs. However, this LSTM system in practice produced higher WERs than the baseline LSTM system trained on tandem features shown in the second line of Table 4, and therefore was not used in our experiments.

9. REFERENCES

[1] Steve Renals, Nelson Morgan, Hervé Bourlard, Michael Cohen, and Horacio Franco, "Connectionist probability estimators in hmm speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, pp. 161–174, 1994.

[2] Herve A Bourlard and Nelson Morgan, *Connectionist speech recognition: a hybrid approach*, vol. 247, Springer Science & Business Media, 2012.

[3] Tony Robinson and Frank Fallside, "A recurrent error propagation network speech recognition system," *Computer Speech and Language*, vol. 5, no. 3, 1991.

[4] Alex Waibel, "Consonant recognition by modular construction of large phonemic time-delay neural networks," in *Advances in neural information processing systems*, 1989, pp. 215–223.

[5] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara 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.

[6] Kyu J Han, Seongjun Hahm, Byung-Hak Kim, Jungsuk Kim, and Ian Lane, "Deep learning-based telephony speech recognition in the wild," in *Proc. Interspeech*, 2017, pp. 1323–1327.

[7] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke, "The microsoft 2017 conversational speech recognition system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5934–5938.

[8] William Hartmann, Roger Hsiao, Tim Ng, Jeff Ma, Francis Keith, and Man-Hung Siu, "Improved single system conversational telephone speech recognition with vgg bottleneck features," *Proc. Interspeech 2017*, pp. 112–116, 2017.

[9] Mingkun Huang, Yongbin You, Zhehuai Chen, Yanmin Qian, and Kai Yu, "Knowledge distillation for sequence model," *Proc. Interspeech 2018*, pp. 3703–3707, 2018.

[10] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[12] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

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

[15] David Barber and Christopher M Bishop, "Ensemble learning in bayesian neural networks," *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, vol. 168, pp. 215–238, 1998.

[16] Alex Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, 2011, pp. 2348–2356.

[17] Jen-Tzung Chien and Yuan-Chu Ku, "Bayesian recurrent neural network for language modeling," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 2, pp. 361–374, 2016.

[18] Franco Manessi and Alessandro Rozza, "Learning combinations of activation functions," *arXiv preprint arXiv:1801.09403*, 2018.

[19] Max WY Lam, Shoukang Hu, Xurong Xie, Shansong Liu, Jianwei Yu, Rongfeng Su, Xunying Liu, and Helen Meng, "Gaussian process neural networks for speech recognition," *Proc. Interspeech 2018*, pp. 1778–1782, 2018.

[20] Pawel Swietojanski, Arnab Ghoshal, and Steve Renals, "Revisiting hybrid and gmm-hmm system combination techniques," in *Proceedings of the IEEE International Conference on*. Wydawnictwo Naukowe PWN-Polish Scientific Publishers PWN, 2013, vol. 21, pp. 1120–1124.

[21] Haipeng Wang, Anton Ragni, Mark John Gales, Katherine Mary Knill, Philip Charles Woodland, and Chao Zhang, "Joint decoding of tandem and hybrid systems for improved keyword spotting on low resource languages," 2015.

[22] Radford M Neal, *Bayesian learning for neural networks*, vol. 118, Springer Science & Business Media, 2012.

[23] Matthias Seeger, "Gaussian processes for machine learning," *International journal of neural systems*, vol. 14, no. 02, pp. 69–106, 2004.

[24] Max WY Lam, Xie Chen, Shoukang Hu, Jianwei Yu, Xunying Su, and Helen Meng, "Gaussian process lstm recurrent neural network language models for speech recognition," Submission to ICASSP 2019.

[25] David M Blei, Alp Kucukelbir, and Jon D McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[26] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al., "The htk book," *Cambridge university engineering department*, vol. 3, pp. 175, 2002.

[27] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[28] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks.," in *Interspeech*, 2013, pp. 2345–2349.

[29] Frantisek Grezl and Petr Fousek, "Optimizing bottle-neck features for lvcsr.," in *ICASSP*, 2008, vol. 8, pp. 4729–4732.

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.