Bayesian x-vector: Bayesian Neural Network based x-vector System for Speaker Verification

Xu Li[⋆]*, Jinghua Zhong*[⋆,†]*, Jianwei Yu*[⋆]*, Shoukang Hu*[⋆]*, Xixin Wu*[⋆]*, Xunying Liu*[⋆]*, Helen Meng*[⋆]

[⋆]Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong
[†]SpeechX Limited, Shenzhen, China
{xuli, jwyu, skhu, wuxx, xyliu, hmmeng}@se.cuhk.edu.hk, jhzhong@speechx.cn

Abstract

Speaker verification systems usually suffer from the mismatch problem between training and evaluation data, such as speaker population mismatch, the channel and environment variations. In order to address this issue, it requires the system to have good generalization ability on unseen data. In this work, we incorporate Bayesian neural networks (BNNs) into the deep neural network (DNN) x-vector speaker verification system to improve the system's generalization ability. With the weight uncertainty modeling provided by BNNs, we expect the system could generalize better on the evaluation data and make verification decisions more accurately. Our experiment results indicate that the DNN x-vector system could benefit from BNNs especially when the mismatch problem is severe for evaluations using out-of-domain data. Specifically, results show that the system could benefit from BNNs by a relative EER decrease of 2.66% and 2.32% respectively for short- and long-utterance in-domain evaluations. Additionally, the fusion of DNN x-vector and Bayesian x-vector systems could achieve further improvement. Moreover, experiments conducted by out-of-domain evaluations, e.g. models trained on Voxceleb1 while evaluated on NIST SRE10 core test, suggest that BNNs could bring a larger relative EER decrease of around 4.69%.

Index terms— speaker verification, Bayesian neural network, DNN x-vector, uncertainty modelling

1. Introduction

We are observing an ever-increasing use of automatic speaker verification (ASV) systems in our everyday lives. An essential step for verification is to disentangle the speaker information from each spoken utterance and then decisions are made based on the speaker similarity. Through decades of years development, three most representative frameworks have been proposed in this research area. (i) Extending from joint factor analysis [1, 2] that models speaker and channel subspaces separately, i-vector based speaker embedding is proposed to jointly model speaker and channel variations together with a speaker-discriminative back-end for decisions. Such systems include Gaussian mixture model (GMM) i-vector [3–6] and deep neural network (DNN) i-vector systems [7]. (ii) Benefiting from the powerful discrimination ability of DNNs, DNN-based speaker embedding is proposed to extract speaker-discriminative representations for each utterance, which could perform as the state of the art on short-utterance evaluation conditions. These systems include d-vectors [8] and x-vectors [9, 10]. (iii) With the development of end-to-end techniques, many researches focus on constructing ASV systems in an end-to-end manner [11–13], which directly learns a mapping from enrollment and testing utterance pairs to verification scores, resulting in a compact structure and comparably good performance.

A challenging issue for ASV systems development is the mismatch between the training and evaluation data, such as the speaker population mismatch, and variations in channel and environmental background. The speaker population used for training and evaluation commonly have no overlap especially for practical applications. To overcome this mismatch usually requires the extracted speaker representations to generalize well on unseen speaker data. The channel and environment variations mostly exist in practical applications where the training and evaluation data are collected from different types of recorders and environments. These mismatches also have a high demand for the model's generalizability on unseen data.

To address this issue, previous efforts [14–16] have applied adversarial training to alleviate the channel and environment variations from utterance embedding. It is achieved by adding adversarial penalty on domain-related information in embedding vectors during the extractor training stage. This approach has been proven to be effective in alleviating the effects of channel and environmental mismatches. However, it does not consider the speaker population mismatch that could also lead to the system performance degradation. In this work, we try to improve the system's generalizability across these kinds of mismatches in a unified approach. Inspired by previous work based on Bayesian learning [17–19], we focus on the DNN x-vector system and apply Bayesian neural networks (BNNs) to improve the x-vector system's generalization ability.

The Bayesian learning approach has been shown to be effective to improve the generalization ability of discriminative training in DNN systems. In the machine learning community, similar work have been conducted to incorporate Bayesian learning into DNN systems. Barber et al. [20] proposed an efficient variational inference strategy for BNNs. Blundell et al. [21] proposed a novel backpropagation-compatible algorithm for learning the network parameters' posterior distribution. In the speech area, some previous work involved BNNs in speech recognition [17, 18, 22, 23]. Especially, Xie et al. [17] proposed the Bayesian learning of hidden unit contributions (BLHUC) for speaker adaptation. The BLHUC could model the uncertainty of speaker-dependent parameters and improve the speech recognition performance especially when given very limited speaker adapatation data. Other work also applied the Bayesian technique into language modelling [19, 24].

In a DNN x-vector system, the parameters of traditional time delay neural network (TDNN) layers estimated via the maximum likelihood strategy are deterministic and tend to over-

fit when given limited training data or when there is a large mismatch between the training and evaluation data. In the case of mismatch in speaker population, the overfitted model parameters may result in speaker representations following a spike distribution towards possible training speaker identities. However this will tend not to generalize well on unseen speaker data. To address this issue, BNNs could help smooth the distributions of speaker representation for better generalization on unseen speaker data.

The cases of channel and environmental mismatch are similar. For instance, for channel mismatch, the overfitted model parameters may partially rely on channel information to classify speakers due to various recorders for different speakers in the training data. However, when generalizing to channel-mismatched evaluation data, the original channel-speaker relationship is broken and the trained reliance on channel information cloud lead to misclassification. To alleviate this issue, BNNs change deterministic parameters to be probabilistic via a posterior distribution. This parameter distribution modeling could reduce the risk of overfitting on channel information by smoothing parameters to consider extra possible values that do not rely on channel information for speaker classification.

The above issues motivate this work to incorporate BNNs into the x-vector system by replacing the TDNN layers. We adopt an efficient variational inference based approach to approximate the parameter posterior distribution. The effectiveness of Bayesian learning is investigated on both short- and long-utterance in-domain evaluation, and also an out-of-domain evaluation that includes larger channel and environment mismatches. To the best of our knowledge, this is the first work that applies Bayesian learning technique to speaker verification systems.

Our experiments are based on Voxceleb1 (for short-utterance condition) and NIST Speaker Recognition Evaluation (SRE) 10 (for long-utterance condition) datasets.

The rest of this paper is organized as follows: Section 2 introduces the baseline system architecture, and the Bayesian neural networks will be illustrated in Section 3. Section 4 states the experimental setups, and Section 5 shows the experimental results. Finally, Section 6 concludes the paper.

2. Baseline: DNN x-vector system

A DNN x-vector system [10] consists of two parts: a front-end used for extracting utterance-level speaker embeddings and a verification scoring back-end. The front-end compresses speech utterances of different length into fixed-dimension speaker-related embeddings (x-vectors). Based on these embeddings, different scoring schemes can be used for judging whether two utterances belong to a same person or not. In this work, we focus on the reversion of the front-end, and choose different back-ends for the performance evaluation.

The x-vector extractor is a neural network trained via a speaker discrimination task, the architecture of which is shown in Fig. 1. It consists of frame-level and utterance-level extractors. At the frame level, several layers of time delay neural network (TDNN) are used to model the time-invariant characteristics of acoustic features. Then the statistics pooling layer aggregates all the frame-level outputs from the last TDNN layer, and computes their mean and standard deviation. The mean and standard deviation are concatenated together and propagated through several fully connected utterance-level layers, i.e. embedding layers, and finally the softmax output layer. The cross-entropy between one-hot speaker labels and the softmax outputs

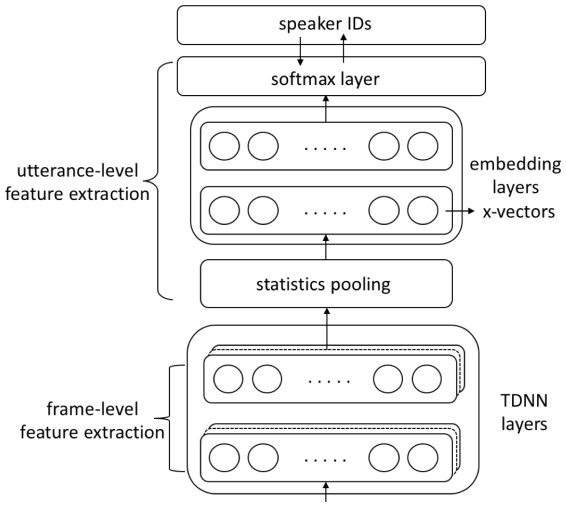

Figure 1: The architecture of a DNN x-vector extractor.

is used as the loss function during the training stage. In the testing stage, given the acoustic features of an utterance, the embedding layer output is extracted as the x-vector. Since the network is trained in a speaker-discriminative manner, the extracted x-vectors are expected to only contain speaker-related information. But in practice, as investigated in [25], x-vectors still contain other speaker-unrelated information, such as channel, transcription and utterance-length. These information could affect the verification performance especially on the mismatched evaluation data.

3. Bayesian neural network

3.1. Weight uncertainty

Traditional neural networks learn a set of deterministic parameters to fit with the training data via the maximum likelihood estimation, and then make inference based on these fixed parameters in the testing stage. This estimation may lead to overconfident parameters when the training data is limited or when there exists a mismatch between the training and evaluation data.

To alleviate this issue, Bayesian neural networks learn the parameters' posterior distribution instead. The posterior distribution $p(w|\mathcal{D})$ based on the training data $\mathcal{D}$ models weight uncertainty and theoretically enables an infinite number of possible model parameters to fit with the data. This weight uncertainty modeling can smooth model parameters and make the model generalize well on unseen data. During the testing stage, the model computes the output $\hat{y}$ given the input x by making an expectation over the weight posterior distribution $P(w|\mathcal{D})$, as shown in Eq. 1.

$$\begin{aligned} p(\hat{y}|x) &= E_{p(w|\mathcal{D})}[p(\hat{y}|x,w)] \\ &= \int p(\hat{y}|x,w)p(w|\mathcal{D})dw \end{aligned} \tag{1}$$

The estimation of the weight posterior distribution $P(w|\mathcal{D})$ is an essential procedure when training a Bayesian neural network. However, the direct estimation is intractable for neural networks of any practical size, since the number of possible weight values could be infinity. So the variational approxima-

tion [20] is commonly adopted to estimate the weight posterior distribution.

3.2. Variational approximation for Bayesian learning

The variational approximation estimates a set of parameters θ_q for a distribution $q(w;\theta_q)$ to approximate the posterior distribution $p(w|\mathcal{D})$. This is achieved by minimizing the Kullback-Leibler (KL) divergence between these two distributions, as shown in Eq. 2.

$$\begin{aligned}\theta_q^\star &= \arg\min_{\theta_q} KL(q(w;\theta_q)||p(w|\mathcal{D})) \\ &= \arg\min_{\theta_q} \int q(w;\theta_q)\log\frac{q(w;\theta_q)}{p(w|\mathcal{D})}dw \quad (2)\\ &= \arg\min_{\theta_q} \int q(w;\theta_q)\log\frac{q(w;\theta_q)}{p(w)p(\mathcal{D}|w)}dw + \log p(\mathcal{D}) \quad (3)\\ &= \arg\min_{\theta_q} \int q(w;\theta_q)\log\frac{q(w;\theta_q)}{p(w)p(\mathcal{D}|w)}dw \quad (4)\\ &= \arg\min_{\theta_q} \int q(w;\theta_q)\log\frac{q(w;\theta_q)}{p(w)}dw \\ &\quad - \int q(w;\theta_q)\log p(\mathcal{D}|w)dw \quad (5)\\ &= \arg\min_{\theta_q} KL(q(w;\theta_q)||p(w)) - E_{q(w;\theta_q)}[\log(p(\mathcal{D}|w))] \quad (6)\end{aligned}$$

From Eq. 2 to 4, we apply Bayes' Rule and drop the constant term $\log p(\mathcal{D})$ that does not affect the minimization over θ_q. Equations from 4 to 6 demonstrate that this minimization equation could be decomposed into two parts: 1) the KL divergence between the approximation distribution $q(w;\theta_q)$ and the prior distribution $p(w)$ on the weight, 2) the expectation of the log likelihood of the training data over the approximation distribution $q(w;\theta_q)$. Eq. 6 is used as the loss function to be minimized during the training process.

As commonly adopted in [20, 21], we assume that the variational approximation follows a diagonal Gaussian distribution with a parameter set $\theta_q = \{\mu_q, \rho_q\}$. μ_q is the mean of the diagonal Gaussian distribution, while ρ_q generates the diagonal Gaussian standard deviation σ_q by $\sigma_q = \log(1+\exp(\rho_q))$. The prior distribution is also assumed to be a diagonal Gaussian distribution with a parameter set $\theta_p = \{\mu_p, \sigma_p\}$. Unlike θ_q will be updated during the training stage, θ_p is usually a set of predetermined fixed parameters.

Under the Gaussian distribution assumptions, the first part in Eq. 6 has a closed-form result that can be computed directly,

$$\begin{aligned}&KL(q(w;\theta_q)||p(w)) \\ &= \sum_{d=1}^{D}[\log(\frac{\sigma_{p,d}}{\sigma_{q,d}}) + \frac{(\mu_{q,d}-\mu_{p,d})^2+\sigma_{q,d}^2}{2\sigma_{p,d}^2} - \frac{1}{2}] \quad (7)\end{aligned}$$

where D denotes the number of entries in the weight matrix, and $\mu_{q,d}$, $\sigma_{q,d}$, $\mu_{p,d}$ and $\sigma_{p,d}$ are the d-th entry of μ_q, σ_q, μ_p and σ_p, respectively.

While the integration in the second part cannot be computed directly, Monte Carlo sampling is commonly applied to approximate this integration, as shown in Eq. 8:

Layer	Layer Context	Input × Output
frame1	[t-2, t+2]	120 × 512
frame2	{t-2, t, t+2}	1536 × 512
frame3	{t-3, t, t+3}	1536 × 512
frame4	{t}	512 × 512
frame5	{t}	512 × 1500
stats pooling	[0, T)	1500T × 3000
segment6	{0}	3000 × 512
segment7	{0}	512 × 512
softmax	{0}	512 × N

Table 1: The x-vector extractor architecture. The N in the softmax layer is the size of speaker population during training. The t in frame-level layers represents the current frame, and T represents the total number of frames in an utterance. x-vectors are extracted from segment6, before the nonlinearity.

$$\begin{aligned}E_{q(w;\theta_q)}[\log(p(\mathcal{D}|w))] &= \int q(w;\theta_q)\log(p(\mathcal{D}|w))dw \\ &\approx \frac{1}{J}\sum_{j=1}^{J}\log(p(\mathcal{D}|w^j)) \quad (8)\end{aligned}$$

where J is the number of samples and $w^j = \mu_q + \sigma_q \bigodot \epsilon_j$ is the j th sample from the distribution $q(w;\theta_q)$, and $\bigodot$ denotes the element-wise multiplication. As the equation shows, w^j is sampled by scaling and shifting a random signal $\epsilon_j \sim N(0, I)$ from the unit Gaussian distribution.

Finally, the loss function is derived as:

$$\begin{aligned}L &= KL(q(w;\theta_q)||p(w)) - E_{q(w;\theta_q)}[\log(p(\mathcal{D}|w))] \\ &\approx \sum_{d=1}^{D}[\log(\frac{\sigma_{p,j}}{\sigma_{q,j}}) + \frac{(\mu_{q,j}-\mu_{p,j})^2+\sigma_{q,j}^2}{2\sigma_{p,j}^2} - \frac{1}{2}] \\ &\quad - \frac{1}{J}\sum_{j=1}^{J}\log(p(\mathcal{D}|w^j)) \quad (9)\end{aligned}$$

The gradient with respective to parameters $\theta_q = \{\mu_q, \rho_q\}$ can be derived as,

$$\frac{\partial L}{\partial \mu_{q,d}} = \frac{\mu_{q,d}-\mu_{p,d}}{\sigma_{p,d}^2} + \frac{1}{J}\sum_{j=1}^{J}G_{j,d} \quad (10)$$

$$\frac{\partial L}{\partial \rho_{q,d}} = [\frac{\sigma_{q,d}}{\sigma_{p,d}^2} - \frac{1}{\sigma_{q,d}}]\frac{e^{\rho_{q,d}}}{1+e^{\rho_{q,d}}} + \frac{1}{J}\sum_{j=1}^{J}\epsilon_{j,d}G_{j,d}\frac{e^{\rho_{q,d}}}{1+e^{\rho_{q,d}}} \quad (11)$$

where $G_{j,d} = -\frac{\partial \log(p(\mathcal{D}|w^j))}{\partial w_d^j}$ is the standard gradient of loss function with respective to w_d^j (the d-th entry of the weight matrix).

4. Experimental setup

In order to evaluate the effectiveness of Bayesian learning for speaker verification in both short- and long-utterance conditions, we perform experiments on two datasets. For the short-utterance condition, we consider the Voxceleb1 [26] dataset, where the recordings are short clips of human speech. There

Table 2: In-domain evaluation: equal error rate (EER) and minimum detection cost function (min-DCF) in different conditions.

Training set	Evaluation set	System	Scoring back-end	x-vector extractor	EER(%)	DCF_{VOX}/DCF_{SRE10}
Voxceleb1	Voxceleb1	(1)	cosine	baseline	9.58	0.6899
		(2)		proposed	9.30	0.6508
		(3)		fusion	**8.64**	**0.6423**
		(4)	PLDA	baseline	6.68	0.6023
		(5)		proposed	6.52	**0.5423**
		(6)		fusion	**6.35**	0.5487
NIST SRE10	NIST SRE10	(7)	cosine	baseline	5.61	0.6830
		(8)		proposed	5.52	0.6555
		(9)		fusion	**5.47**	**0.6502**
		(10)	PLDA	baseline	3.29	0.3926
		(11)		proposed	3.19	**0.3835**
		(12)		fusion	**3.17**	0.3840

are totally 148,642 utterances for 1,251 celebrities. We follow the configuration in [26], where 4,874 utterances from 40 speakers are reserved for evaluation, and the remaining utterances are used for training x-vector systems (the baseline and BNN-based system) and the back-end model parameters.

For the long-utterance condition, the core test in the NIST speaker recognition evaluation 10 (SRE10) [27] is used for evaluation, where the recordings are long-duration telephone speech. The training data used in this condition includes SREs from 04 through 08, Switchboard2 Phases 1, 2 and 3, and Switchboard cellular. The Switchboard portion is commonly used to increase the training data variety [9, 10]. In total there are around 65,000 recordings for 6,500 speakers. All this training data is used for training x-vector systems (the baseline and BNN-based system), while only the SRE parts are used for training the scoring back-ends. During the training stage, since the utterances are very long, the GPU memories limitation forces a tradeoff between minibatch size and maximum training example length. We randomly cut each recording into chunks of length from 2s to 10s (200 frames to 1000 frames) along with a minibatch size of 48. After this procedure, the average training chunks for each speaker is around 220. No data augmentation technique is applied in any experiment.

Mel-frequency cepstral coefficients (MFCCs) are adopted as acoustic features in all the experiments. Before extracting MFCCs, a pre-emphasis with coefficient of 0.97 is adopted. "Hamming" window having size of 25ms and step-size of 10ms is applied to extract a frame, and finally 30 cepstral coefficients are kept. The extracted MFCCs are mean-normalized over a sliding window of up to 3 seconds, and voice activity detection (VAD) [9] filters out nonspeech frames.

The configuration of the baseline x-vector extractor is consistent with [10], as shown in Table 1. After propagating through several frame-level layers and one statistic pooling layer, the outputs from the first segment-level layer (segment6 in the table) before nonlinearity are extracted as x-vectors. We adopt the stochastic gradient descent (SGD) optimizer during the training stage. In order to make a fair comparison, the Bayesian x-vector system is configured with the same architecture of the baseline system except the first TDNN layer is replaced by BNN layer with the same number of units. We also attempted to replace other layers with BNN layer, but experiment results show that operation on other layers gives a slightly worse performance than operation on the first layer. This may indicate that operation on the layer close to the input features is more effective and has more impact on improving the model's generalization ability. The choice of prior distribution has an impact on model convergence and training efficiency. In this work, we set the prior distribution based on the baseline model parameters, similar with the strategy in [18]. The x-vector systems (the baseline and BNN-based system) are implemented by Pytorch [28], while the other parts, including data preparation, feature extraction and training scoring back-ends, are implemented by Kaldi toolkit [29].

To evaluate the generalization benefits that Bayesian learning could bring under evaluation of different mismatch degrees, we design two kinds of evaluation experiments: in-domain evaluation and out-of-domain evaluation. The training and testing stages are executed on the same dataset in in-domain evaluation, while they are executed on different datasets in out-of-domain evaluation. On both evaluations, we perform experiments on two datasets, i.e. Voxceleb1 for the short-utterance condition and NIST SRE10 for the long-utterance condition. Two kinds of scoring back-ends are adopted in our experiments: cosine and probabilistic linear discriminative analysis (PLDA) back-ends. Before propagating into the back-end scoring, speaker embeddings are projected into a lower dimension space via linear discriminant analysis (LDA). Following the default settings adopted in Kaldi toolkit [29], the LDA dimension is set as 150 and 200 for cosine and PLDA scoring, respectively.

The evaluation metrics adopted in this work are the commonly used equal error rate (EER) and minimum detection cost function (min-DCF). For the min-DCF metric, we consider the prior probability of target trials to be 0.01 on the Voxceleb1 (denoted as DCF_{VOX}). For the NIST SRE10 dataset, the target trial partitions are much smaller and thus we consider the prior probability to be 0.001 (denoted as DCF_{SRE10}). Intuitively, with the consideration that the baseline system could have correct operation with high confidence on the common characterisitcs between the training and evaluation data, while the BNN-based system could generate well on the mismatch characteristics, we also design a fusion system for performance comparison. The fusion is operated by averaging the verification scores from the two systems.

5. Experiment results

5.1. In-domain evaluation

In this section, we perform in-domain evaluation on two datasets: Voxceleb1 for the short-utterance condtion and NIST SRE10 for the long-utterance condition. The corresponding performance is shown in Table 2. From the table, we ob-

Table 3: Out-of-domain evaluation: equal error rate (EER) and minimum detection cost function (min-DCF) in different conditions.

Training set	Evaluation set	System	Scoring back-end	x-vector extractor	EER(%)	DCF_{VOX}/DCF_{SRE10}
Voxceleb1	NIST SRE10	(1)	cosine	baseline	10.78	0.8650
		(2)		proposed	10.38	0.8633
		(3)		fusion	**10.15**	**0.8428**
		(4)	PLDA	baseline	8.31	0.8646
		(5)		proposed	7.84	0.8541
		(6)		fusion	**7.71**	**0.8378**
NIST SRE10	Voxceleb1	(7)	cosine	baseline	15.30	0.8101
		(8)		proposed	14.85	0.8164
		(9)		fusion	**14.14**	**0.7913**
		(10)	PLDA	baseline	11.27	0.7636
		(11)		proposed	10.91	0.7555
		(12)		fusion	**10.68**	**0.7461**

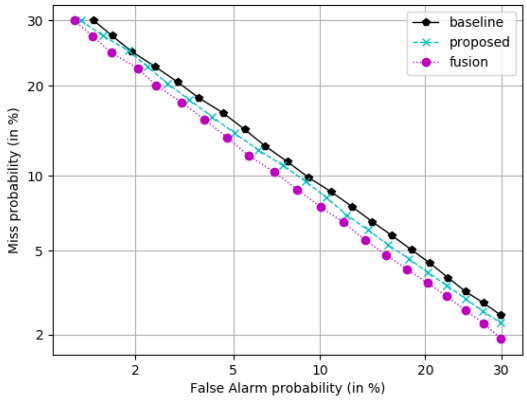

Figure 2: In-domain evaluation: detection error trade-off (DET) curves on the Voxceleb1 dataset, using a cosine scoring back-end.

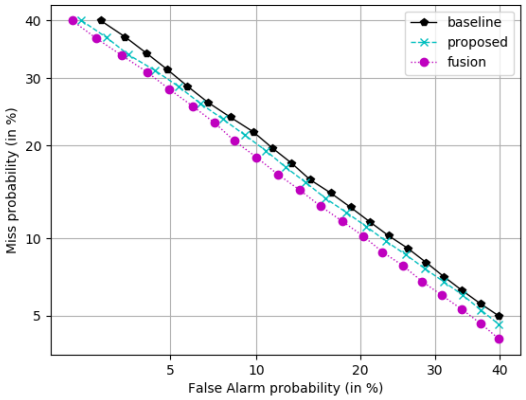

Figure 3: Out-of-domain evaluation: detection error trade-off (DET) curves on the Voxceleb1 dataset, using a cosine scoring backend.

serve that EERs consistently decrease after incorporating the Bayesian learning in both short- and long-utterance conditions. In each condition, we consider the average relative EER decrease across cosine and PLDA back-ends. In the short-utterance condition, the average relative EER decrease from Bayesian x-vector system is 2.66%, and the fusion system could achieve further average relative EER decrease by 7.24%. For the long-utterance condition, the average relative EER decrease is 2.32% for Bayesian x-vector system and 3.08% for the fusion system.

Our experiment results show that systems in the short-utterance condition could benefit more from the Bayesian learning when compared with the long-utterance condition. One possible explanation is that, in the short-utterance condition, systems may be heavily affected by speaker-unrelated information, such as channel and phonetic information, which may bring larger mismatches in the testing stage. With the uncertainty modeling of model parameters, the Bayesian learning could bring extra benefits to alleviate these larger mismatches and improve the performance. Similar results could be observed in detection cost function (DCF) metrics as shown in the last column of Table 2. Fig. 2 illustrates the detection error trade-off (DET) curves of systems with the cosine back-end (Systems 1, 2 and 3 in Table 2). It shows that the proposed Bayesian system outperforms the baseline for all operating points, and the fusion system could achieve further improvements due to the complementary advantages of the baseline and the Bayesian system.

5.2. Out-of-domain evaluation

The out-of-domain evaluation is performed in this section, as shown in Table 3. The model trained on Voxceleb1 (Systems 1 to 6) will be evaluated on NIST SRE10, and vice versa. System performance usually drops significantly due to the larger mismatch between the training and evaluation data, so this evaluation has a higher demand for the system's generalization ability.

From Table 3, we observe that systems could benefit more from the generalization power of Bayesian learning. We also consider the average relative EER decrease across cosine and PLDA scoring back-ends for performance evaluation. In the experiments evaluated on NIST SRE10, the average relative EER decrease is 4.69% and 6.53% for the Bayesian system and the fusion system, respectively. For the experiments on the Voxceleb1 dataset, the average relative EER decrease is 3.07% for the Bayesian x-vector system, and the fusion system achieves a further average relative EER decrease of 6.41%. The larger relative EER decrease compared with that in in-domain evaluation suggests that Bayesian learning could be more beneficial

when larger mismatch exists between the training and evaluation data. This phenomenon is similar to the observations stated in [23], where the improvement on the out-of-domain dataset (with larger mismatch) is larger than the in-domain dataset. The last column in Table 3 shows the corresponding DCF performance, and we observe consistent improvement by applying Bayesian learning and the fusion system. Similar to the observations in Fig. 2, the DET curves in Fig. 3 show consistent improvements by applying Bayesian learning and the fusion model for all operating points.

6. Conclusion

In this work, we incorporate the BNN technique into the DNN x-vector system to improve the model's generalization ability. BNN layers embedded in the x-vector extractor make the extracted speaker embedding (Bayesian x-vector) generalize better on unseen data. Our experimental results show that the DNN x-vector could benefit from Bayesian learning for both short- and long-utterance conditions, and the fusion system could achieve further performance improvements. Moreover, we observe that systems could benefit more from Bayesian learning in out-of-domain evaluation. Especially, in out-of-domain evaluation performed on the NIST SRE10 dataset, the average relative EER decrease across cosine and PLDA scoring is around 4.69% and 6.53% by applying the Bayesian system and the fusion system, respectively. This suggests that Bayesian learning is more beneficial when larger mismatch exists between the training and evaluation data. Possible future research will focus on incorporating Bayesian learning into the end-to-end speaker verification systems.

7. Acknowledgements

This work is partially supported by a grant from the HKSAR Government's Research Grants Council General Research Fund (reference number 14208718).

8. References

[1] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[2] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Speaker and session variability in GMM-based speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1448–1460, 2007.

[3] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[4] P. Kenny, "A small footprint i-vector extractor," in *The Speaker and Language Recognition Workshop*, 2012.

[5] S. Prince and J. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[6] D. Garcia-Romero and C. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[7] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *ICASSP*. IEEE, 2014, pp. 1695–1699.

[8] E. Variani, X. Lei, E. McDermott, I. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *ICASSP*. IEEE, 2014, pp. 4052–4056.

[9] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification.," in *Interspeech*, 2017, pp. 999–1003.

[10] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *ICASSP*. IEEE, 2018, pp. 5329–5333.

[11] S. Zhang, Z. Chen, Y. Zhao, J. Li, and Y. Gong, "End-to-end attention based text-dependent speaker verification," in *Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 171–178.

[12] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *ICASSP*. IEEE, 2016, pp. 5115–5119.

[13] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 165–170.

[14] Q. Wang, W. Rao, S. Sun, L. Xie, E. S. Chng, and H. Li, "Unsupervised domain adaptation via domain adversarial training for speaker recognition," in *ICASSP*. IEEE, 2018, pp. 4889–4893.

[15] G. Bhattacharya, J. Alam, and P. Kenny, "Adapting end-to-end neural speaker verification to new languages and recording conditions with adversarial training," in *ICASSP*. IEEE, 2019, pp. 6041–6045.

[16] Y. Tu, M. Mak, and J. Chien, "Variational domain adversarial learning for speaker verification," in *Interspeech*, 2019, pp. 4315–4319.

[17] X. Xie, X. Liu, T. Lee, S. Hu, and L. Wang, "Blhuc: Bayesian learning of hidden unit contributions for deep neural network speaker adaptation," in *ICASSP*. IEEE, 2019, pp. 5711–5715.

[18] S. Hu, M. Lam, X. Xie, S. Liu, J. Yu, X. Wu, X. Liu, and H. Meng, "Bayesian and gaussian process neural networks for large vocabulary continuous speech recognition," in *ICASSP*. IEEE, 2019, pp. 6555–6559.

[19] J. Yu, M. WY Lam, S. Hu, X. Wu, X. Li, Y. Cao, X. Liu, and H. Meng, "Comparative study of parametric and representation uncertainty modeling for recurrent neural network language models," in *Interspeech*, 2019, pp. 3510–3514.

[20] D. Barber and C. M Bishop, "Ensemble learning in Bayesian neural networks," *Nato ASI Series F Computer and Systems Sciences*, vol. 168, pp. 215–238, 1998.

[21] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

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

[23] S. Hu, X. Xie, S. Liu, M. WY Lam, J. Yu, X. Wu, X. Liu, and H. Meng, "LF-MMI training of Bayesian and Gaussian process time delay neural networks for speech recognition," in *Interspeech*, 2019, pp. 2793–2797.

[24] J. Chien and Y. Ku, "Bayesian recurrent neural network for language modeling," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 2, pp. 361–374, 2015.

[25] D. Raj, D. Snyder, D. Povey, and S. Khudanpur, "Probing the information encoded in x-vectors," *arXiv preprint arXiv:1909.06351*, 2019.

[26] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[27] A. F Martin and C. S Greenberg, "The NIST 2010 speaker recognition evaluation," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.

[29] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *Workshop on Automatic Speech Recognition and Understanding*, 2011.