Discriminative Feature Extraction Based on Sequential Variational Autoencoder for Speaker Recognition

Takenori Yoshimura*, Natsumi Koike*, Kei Hashimoto*, Keiichiro Oura*,
Yoshihiko Nankaku*, and Keiichi Tokuda*
* Nagoya Institute of Technology, Nagoya, Japan
E-mail: {takenori,koike.n,bonanza,uratec,nankaku,tokuda}@sp.nitech.ac.jp

***Abstract*—This paper presents an extended version of the variational autoencoder (VAE) for sequence modeling. In contrast to the original VAE, the proposed model can directly handle variable-length observation sequences. Furthermore, the discriminative model and the generative model are simultaneously learned in a unified framework. The network architecture of the proposed model is inspired by the i-vector/PLDA framework, whose effectiveness has been proven in sequence modeling tasks such as speaker recognition. Experimental results on the TIMIT database show that the proposed model outperforms the traditional i-vector/PLDA system.**

I. Introduction

To perform sequence labeling tasks such as speaker recognition, emotion recognition, and language identification, many approaches have been proposed. The i-vector framework [1], [2], [3], [4] is one of the most successful ways to extract a compact representation of a variable-length sequence, especially in speaker verification. In this approach, high-dimensional vectors formed through the concatenation of the adapted Gaussian mixture model (GMM) mean vectors [5] are transformed into low-dimensional total variability components using a factor analysis technique. A set of the extracted low-dimensional factors so-called i-vectors are then modeled by a probabilistic linear discriminant analysis (PLDA) model. Although the effectiveness of the i-vector based approach has been proven in various applications, it is expected that deep learning techniques enable the extracttion of more compact and discriminative representations from observation sequences.

Some attempts are being made to learn an embedding space using deep neural networks (DNNs). In the d-vector based approach [6], [7], a DNN with a softmax output layer is trained to map a frame-level observation to the corresponding class label. A compact discriminative representation is computed as the statistics of the activations derived from the last or second last hidden layer of the DNN. In the x-vector based approach [8], [9], a sequence of frame-level observations is fed to a DNN classifier. A fixed-dimensional class-dependent representation is obtained by a statistics pooling layer in the DNN. Instead of the softmax output layer, some of the attempts [10], [11], [12] use triplet loss [13], which minimizes the distance between embedding pairs from the same label and maximizes the distance between pairs from different labels. It can be viewed that the above-mentioned approaches integrate i-vector extraction and PLDA modeling in a unified way.

Recently, an autoencoder and its probabilistic version, a variational autoencoder (VAE) [14], have been used to obtain latent space representations of inputs such as images and spectral envelopes [15]. The main idea of the autoencoder is to learn latent representations automatically from observations in an unsupervised manner using two neural networks: an encoder and a decoder. The encoder compresses an observation into a latent space and the decoder reconstructs the observation given the encoded latent representation. One of the main advantages is that they can be used for semi-supervised learning, which is an important aspect when a small amount of labeled data and a large amount of unlabeled data are available. Thus, it is expected that autoencoder based models can exploit a large collection of given data. However, in the framework of a standard autoencoder, directly handling a *sequence* of observations such as speech is not straightforward. In speaker verification, although a few approaches use a variant of the autoencoder as the PLDA model in the i-vector/PLDA framework [16], [17], the input of the autoencoder is a fixed-dimensional i-vector rather than a variable-length observation sequence.

This paper proposes a VAE based model as an extended version of the i-vector/PLDA framework. In contrast to the original VAE [18], the proposed model can handle variable-length observation sequences and extract frame-level and sequence-level latent representations that are independent of each other. Furthermore, the discriminative model for identifying class labels and the generative model for providing observations are simultaneously trained in a unified framework. This can be viewed as a kind of multi-task learning (MTL) [19].

The remainder of the paper is organized as follows: Section 2 briefly explains the i-vector/PLDA paradigm. Section 3 describes the proposed model based on the VAE. Section 4 reports on experiments on speaker recognition using the TIMIT database. Conclusions and future work are presented in Section 5.

II. I-vector/PLDA

In the i-vector/PLDA framework, given a supervector of mean vectors, $\boldsymbol{\mu}_s \in \mathbb{R}^{DM}$, from an adapted universal background model (UBM) that has D-dimensional M Gaussian components, a class-specific supervector $\boldsymbol{s} \in \mathbb{R}^{DM}$ is assumed to be

$$\boldsymbol{s} = \boldsymbol{\mu}_s + \boldsymbol{T}\boldsymbol{x}, \tag{1}$$

where $\boldsymbol{T} \in \mathbb{R}^{DM \times R'}$ is a low-rank matrix whose columns span the total variability in the supervector space ($R' \ll DM$), and $\boldsymbol{x} \in \mathbb{R}^{R'}$ is a latent vector following a standard Gaussian distribution. The i-vector representation of a data sequence is defined as the mean of the posterior distribution of $\boldsymbol{x}$. The extracted i-vector $\boldsymbol{x}$ is then modeled by assuming a factor analysis model so-called PLDA:

$$\boldsymbol{x} = \boldsymbol{\mu}_x + \boldsymbol{\phi}\boldsymbol{z} + \boldsymbol{n}, \tag{2}$$

where $\boldsymbol{\mu}_x \in \mathbb{R}^{R'}$ is the average of the extracted i-vectors, $\boldsymbol{\phi} \in \mathbb{R}^{R' \times R}$ is a factor loading matrix, $\boldsymbol{z} \in \mathbb{R}^{R}$ is a vector of class-dependent latent factors following a standard Gaussian distribution, and $\boldsymbol{n} \in \mathbb{R}^{R'}$ is a full covariance residual noise term that explains the variability not captured through the latent factors.

It is expected that more discriminative representation can be extracted by replacing the multiple-stage extraction to a single DNN based nonlinear feature extraction.

III. Proposed model based on VAE

The basic idea of the proposed model is to extract meaningful latent representations from a sequence of observations using a deep generative model. To achieve that goal, a discriminative model is embedded in the network structure. In addition, the structure of the i-vector/PLDA paradigm is introduced to the sequence-level encoder of our model. It should be noted that because the proposed model is based on the VAE, it can be used for semi-supervised learning.

A. Objective function

A standard VAE consists of an encoder and a decoder, where the encoder compresses an observation $\boldsymbol{o}$ into a low-dimensional latent representation $\boldsymbol{z}$, and the decoder tries to reconstruct the $\boldsymbol{o}$ given the $\boldsymbol{z}$ (see Fig. 1(a)). By conditioning on another description of an observation c, the VAE can be used for semi-supervised learning (see Fig. 1(b)). In the conditional VAE [18], the log-likelihood function of labeled and unlabeled data can be rewritten as follows respectively:

$$\begin{aligned}\log p(\boldsymbol{o}, c) &= \log \int p(\boldsymbol{o} \mid c, \boldsymbol{z})\, p(c)\, p(\boldsymbol{z})\, d\boldsymbol{z} \\ &\geq \int q(\boldsymbol{z} \mid c) \log \frac{p(\boldsymbol{o} \mid c, \boldsymbol{z})\, p(c)\, p(\boldsymbol{z})}{q(\boldsymbol{z} \mid c)} d\boldsymbol{z} \\ &\equiv -\mathcal{L}_{org}\end{aligned} \tag{3}$$

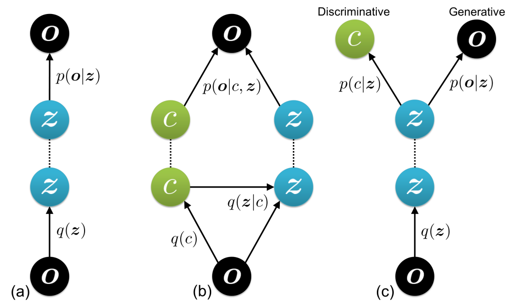

Fig. 1. Graphical representations of (a) the standard VAE, (b) the conditional VAE, and (c) the proposed model.

and

$$\begin{aligned}\log p(\boldsymbol{o}) &= \log \sum_c \int p(\boldsymbol{o} \mid c, \boldsymbol{z})\, p(c)\, p(\boldsymbol{z})\, d\boldsymbol{z} \\ &\geq \sum_c \int q(c)\, q(\boldsymbol{z} \mid c) \log \frac{p(\boldsymbol{o} \mid c, \boldsymbol{z})\, p(c)\, p(\boldsymbol{z})}{q(c)\, q(\boldsymbol{z} \mid c)} d\boldsymbol{z} \\ &\equiv -\mathcal{U}_{org},\end{aligned} \tag{4}$$

where $q(\cdot)$ denotes an approximate posterior distribution $q(\cdot \mid \boldsymbol{o})$. Since the discriminative model $q(c)$ contributes only to (4) relating to the unlabeled data, an additional term related to $q(c)$ is added to the loss function:

$$\mathcal{J} = \mathcal{L}_{org} + \mathcal{U}_{org} - \alpha \log q(c) \tag{5}$$

where α is a hyperparameter that controls the relative weight between generative and discriminative learning.

On the other hand, the proposed model assumes the following log-likelihood functions for labeled and unlabeled data:

$$\begin{aligned}\log p(\boldsymbol{o}, c) &= \log \int p(\boldsymbol{o} \mid \boldsymbol{z})\, p(c \mid \boldsymbol{z})\, p(\boldsymbol{z})\, d\boldsymbol{z} \\ &\geq \int q(\boldsymbol{z}) \log \frac{p(\boldsymbol{o} \mid \boldsymbol{z})\, p(c \mid \boldsymbol{z})\, p(\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z} \\ &\equiv -\mathcal{L}_{our}\end{aligned} \tag{6}$$

and

$$\begin{aligned}\log p(\boldsymbol{o}) &= \log \int p(\boldsymbol{o} \mid \boldsymbol{z})\, p(\boldsymbol{z})\, d\boldsymbol{z} \\ &\geq \int q(\boldsymbol{z}) \log \frac{p(\boldsymbol{o} \mid \boldsymbol{z})\, p(\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z} \\ &\equiv -\mathcal{U}_{our},\end{aligned} \tag{7}$$

respectively. Since the discriminative model is naturally embedded in (6) (see Fig. 1(c)), there is no need to add an additional term to the loss function:

$$\mathcal{J} = \mathcal{L}_{our} + \mathcal{U}_{our}. \tag{8}$$

It can be seen that both the discriminative model and the generative model are consistently learned in an MTL framework.

For modeling a sequence of observations $\boldsymbol{o}_{1:T} \in \mathbb{R}^{D \times T} = (\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T)$ with a sequence of frame-level latent vector $\boldsymbol{z}_{1:T} \in \mathbb{R}^{Q \times T} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_T)$ and a sequence-level latent vector $\boldsymbol{z}^{(c)} \in \mathbb{R}^R$, we define the following objective function according to (6):

$$
\begin{aligned}
\log p\left(\boldsymbol{o}_{1:T}, c\right) &= \log \iint p\left(\boldsymbol{o}_{1:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{z}^{(c)}\right) p\left(c \mid \boldsymbol{z}^{(c)}\right) \\
&\quad \times p\left(\boldsymbol{z}_{1:T}\right) p\left(\boldsymbol{z}^{(c)}\right) d\boldsymbol{z}_{1:T}\, d\boldsymbol{z}^{(c)} \\
&\geq -\left(\mathcal{L}_{dec}^{(\boldsymbol{o})} + \mathcal{L}_{dec}^{(c)} + \mathcal{L}_{kld}^{(\boldsymbol{o})} + \mathcal{L}_{kld}^{(c)}\right), \qquad (9)
\end{aligned}
$$

where

$$
\begin{aligned}
\mathcal{L}_{dec}^{(\boldsymbol{o})} &= -\sum_{t=1}^{T} \iint q\left(\boldsymbol{z}_t\right) q\left(\boldsymbol{z}^{(c)}\right) \\
&\quad \times \log p\left(\boldsymbol{o}_t \mid \boldsymbol{z}_t, \boldsymbol{z}^{(c)}\right) d\boldsymbol{z}_t\, d\boldsymbol{z}^{(c)}, \qquad (10) \\
\mathcal{L}_{dec}^{(c)} &= -\int q\left(\boldsymbol{z}^{(c)}\right) \log p\left(c \mid \boldsymbol{z}^{(c)}\right) d\boldsymbol{z}^{(c)}, \qquad (11) \\
\mathcal{L}_{kld}^{(\boldsymbol{o})} &= \sum_{t=1}^{T} D_{\mathrm{KL}}\left(q\left(\boldsymbol{z}_t\right) \| \, p\left(\boldsymbol{z}_t\right)\right), \qquad (12) \\
\mathcal{L}_{kld}^{(c)} &= D_{\mathrm{KL}}\left(q\left(\boldsymbol{z}^{(c)}\right) \| \, p\left(\boldsymbol{z}^{(c)}\right)\right), \qquad (13)
\end{aligned}
$$

where D_{KL} denotes the Kullback-Leibler divergence between two distributions. A standard Gaussian distribution is used as the prior distribution:

$$
\begin{aligned}
p\left(\boldsymbol{z}_t\right) &= \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right), \qquad (14) \\
p\left(\boldsymbol{z}^{(c)}\right) &= \mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right), \qquad (15)
\end{aligned}
$$

where $\mathbf{0}$ and $\boldsymbol{I}$ are a zero vector and an identity matrix, respectively.

A set of hyperparameters λ_1, λ_2, and λ_3 is introduced to the loss function to control the relative importance of each objective:

$$
\mathcal{J} = \frac{\lambda_1}{DT}\mathcal{L}_{dec}^{(\boldsymbol{o})} + \frac{1}{C}\mathcal{L}_{dec}^{(c)} + \frac{\lambda_2}{QT}\mathcal{L}_{kld}^{(\boldsymbol{o})} + \frac{\lambda_3}{R}\mathcal{L}_{kld}^{(c)}. \qquad (16)
$$

where C is the total number of unique labels. Since (10) and (11) involve intractable integrals over all values of $\boldsymbol{z}_{1:T}$ and $\boldsymbol{z}^{(c)}$, the expectations are approximated by Monte Carlo sampling so that the gradient of the loss function can be computed using the reparameterization trick [14].

B. Network architectures

Fig. 2 illustrates the entire architecture of the proposed model. It mainly consists of four components. The following sections introduce each of the components in detail.

1) Frame-level decoder: For modeling the frame-level observation $\boldsymbol{o}_t$, a multivariate Gaussian is assumed:

$$
p\left(\boldsymbol{o}_t \mid \boldsymbol{z}_t, \boldsymbol{z}^{(c)}\right) = \mathcal{N}\left(\boldsymbol{o}_t \mid \boldsymbol{\mu}_{\boldsymbol{o}_t}, \boldsymbol{\Sigma}_{\boldsymbol{o}_t}\right), \qquad (17)
$$

where

$$
\left\{\boldsymbol{\mu}_{\boldsymbol{o}_t}, \boldsymbol{\Sigma}_{\boldsymbol{o}_t}\right\} = g\left(\boldsymbol{z}_t, \boldsymbol{z}^{(c)}\right), \qquad (18)
$$

and $g(\cdot)$ is a function represented by a neural network. The input of the network is both the class-dependent latent representation $\boldsymbol{z}^{(c)}$ and the residual latent representation $\boldsymbol{z}_t$.

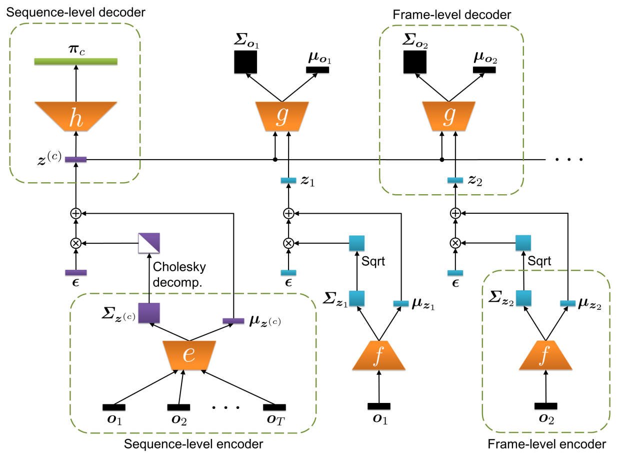
Sequence-level decoder
Frame-level decoder
$\boldsymbol{\pi}_c$
h
$\boldsymbol{z}^{(c)}$
$\boldsymbol{\Sigma}_{\boldsymbol{o}_1}$
$\boldsymbol{\mu}_{\boldsymbol{o}_1}$
g
$\boldsymbol{z}_1$
$\boldsymbol{\Sigma}_{\boldsymbol{o}_2}$
$\boldsymbol{\mu}_{\boldsymbol{o}_2}$
g
$\boldsymbol{z}_2$
ϵ
Cholesky decomp.
Sqrt
$\boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}}$
$\boldsymbol{\mu}_{\boldsymbol{z}^{(c)}}$
e
$\boldsymbol{o}_1$ $\boldsymbol{o}_2$ ... $\boldsymbol{o}_T$
$\boldsymbol{\Sigma}_{\boldsymbol{z}_1}$
$\boldsymbol{\mu}_{\boldsymbol{z}_1}$
f
$\boldsymbol{o}_1$
Sqrt
$\boldsymbol{\Sigma}_{\boldsymbol{z}_2}$
$\boldsymbol{\mu}_{\boldsymbol{z}_2}$
f
$\boldsymbol{o}_2$
Sequence-level encoder
Frame-level encoder

Fig. 2. Entire network architecture of the proposed model, where ϵ is a random number sampled from a standard Gaussian distribution.

2) Sequence-level decoder: A categorical distribution is assumed to model the sequence-level class label c:

$$
p\left(c \mid \boldsymbol{z}^{(c)}\right) = Cat\left(c \mid \boldsymbol{\pi}_c\right), \qquad (19)
$$

where

$$
\boldsymbol{\pi}_c = h\left(\boldsymbol{z}^{(c)}\right), \qquad (20)
$$

and $h(\cdot)$ denotes a neural network with a softmax output layer.

3) Frame-level encoder: For extracting the frame-level latent representation $\boldsymbol{z}_t$, a multivariate Gaussian is used in the same ways as in the frame-level decoder:

$$
q\left(\boldsymbol{z}_t\right) = \mathcal{N}\left(\boldsymbol{z}_t \mid \boldsymbol{\mu}_{\boldsymbol{z}_t}, \boldsymbol{\Sigma}_{\boldsymbol{z}_t}\right), \qquad (21)
$$

where

$$
\left\{\boldsymbol{\mu}_{\boldsymbol{z}_t}, \boldsymbol{\Sigma}_{\boldsymbol{z}_t}\right\} = f\left(\boldsymbol{o}_t\right), \qquad (22)
$$

and $f(\cdot)$ denotes a neural network whose input is the frame-level observation $\boldsymbol{o}_t$.

4) Sequence-level encoder: This is the most important part of the proposed model. The sequence-level feature $\boldsymbol{z}^{(c)}$ is assumed to be sampled from a multivariate Gaussian:

$$
q\left(\boldsymbol{z}^{(c)}\right) = \mathcal{N}\left(\boldsymbol{z}^{(c)} \mid \boldsymbol{\mu}_{\boldsymbol{z}^{(c)}}, \boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}}\right), \qquad (23)
$$

where

$$
\left\{\boldsymbol{\mu}_{\boldsymbol{z}^{(c)}}, \boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}}\right\} = e\left(\boldsymbol{o}_{1:T}\right), \qquad (24)
$$

and $e(\cdot)$ denotes any function. Although there is no restriction about the network architecture of e, we introduce the structure of the i-vector extractor to exploit its powerful capability on sequence modeling. To reformulate the i-vector extractor on the basis of a DNN, a factorized version [20] of a Gaussian mixture model is assumed:

$$\begin{aligned} q\left(\boldsymbol{z}^{(c)}\right) &= C_{\boldsymbol{z}^{(r)}} p\left(\boldsymbol{z}^{(c)}\right) \exp\left\langle \log p\left(\boldsymbol{o}_{1:T} \,|\, \boldsymbol{m}, \boldsymbol{z}^{(c)}\right)\right\rangle_{q(\boldsymbol{m})} \\ &= C_{\boldsymbol{z}^{(r)}} p\left(\boldsymbol{z}^{(c)}\right) \exp\left[\sum_{t=1}^{T}\sum_{m=1}^{M} \gamma_{m,t} \right. \\ &\quad \left. \times \log \mathcal{N}\left(\boldsymbol{o}_t \,\middle|\, \boldsymbol{W}_m \boldsymbol{z}^{(c)} + \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\right)\right] \\ &= \mathcal{N}\left(\boldsymbol{z}^{(c)} \,|\, \boldsymbol{\mu}_{\boldsymbol{z}^{(c)}}, \boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}}\right), \end{aligned} \tag{25}$$

where $C_{\boldsymbol{z}^{(r)}}$ denotes a normalization term, $\boldsymbol{m}$ is a sequence of mixture component indices, $\langle\cdot\rangle_{q(\cdot)}$ represents an expectation with respect to $q(\cdot)$, and $\gamma_{m,t}$ is a posterior probability of being in mixture component m at frame t for a given observation $\boldsymbol{o}_t$. The mean vector and the covariance matrix of the approximate posterior distribution $q\left(\boldsymbol{z}^{(c)}\right)$ can be represented as

$$\boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}} = \left(\boldsymbol{I} + \sum_{t=1}^{T}\sum_{m=1}^{M} \gamma_{m,t} \boldsymbol{W}_m^\top \boldsymbol{\Sigma}_m^{-1} \boldsymbol{W}_m\right)^{-1}, \tag{26}$$

$$\boldsymbol{\mu}_{\boldsymbol{z}^{(c)}} = \boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}} \left(\sum_{t=1}^{T}\sum_{m=1}^{M} \gamma_{m,t} \boldsymbol{W}_m^\top \boldsymbol{\Sigma}_m^{-1} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_m\right)\right). \tag{27}$$

In order to extract more discriminative sequence-level representation from a sequence of observations, some of the terms in (26) and (27) are replaced with nonlinear functions, e_γ and $e_{\boldsymbol{\mu}}$:

$$\begin{aligned} \boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}} &\equiv \left(\boldsymbol{I} + \sum_{t=1}^{T} \underbrace{\sum_{m=1}^{M} e_\gamma^{(m)}(\boldsymbol{o}_t)\, \boldsymbol{L}_m^\top \boldsymbol{L}_m}_{\boldsymbol{P}_t}\right)^{-1} \qquad (28) \\ &= \left(\boldsymbol{I} + \boldsymbol{P}\right)^{-1}, \\ \boldsymbol{\mu}_{\boldsymbol{z}^{(c)}} &\equiv \boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}} \left(\sum_{t=1}^{T} \underbrace{\sum_{m=1}^{M} e_\gamma^{(m)}(\boldsymbol{o}_t) \cdot e_{\boldsymbol{\mu}}^{(m)}(\boldsymbol{o}_t)}_{\boldsymbol{v}_t}\right) \qquad (29) \\ &= \boldsymbol{\Sigma}_{\boldsymbol{z}^{(c)}} \boldsymbol{v}, \end{aligned}$$

where $e_{(\cdot)}^{(m)}$ is an m^{th} output of the function $e_{(\cdot)}$ and $\boldsymbol{L}_m \in \mathbb{R}^{R\times D}$ is trainable parameters. The $\boldsymbol{L}_m$ of course can be replaced by an output of a neural network. Fig. 3 shows the architecture of the sequence-level encoder e.

C. Attention mechanism

It can be said that not all of a sequence of observations contribute to the estimation of sequence-level labels, e.g., the silence frame in speaker recognition. Inspired by recent work on deep learning including attention mechanisms [21], [22] and gate structures [23], [24], an additional network structure is added to the sequence-level encoder. To be precise, the $\boldsymbol{v}_t$ in (29) is rewritten as

$$\boldsymbol{v}_t = e_{\boldsymbol{\sigma}}(\boldsymbol{o}_t) \odot \left(\sum_{m=1}^{M} e_\gamma^{(m)}(\boldsymbol{o}_t) \cdot e_{\boldsymbol{\mu}}^{(m)}(\boldsymbol{o}_t)\right) \tag{30}$$

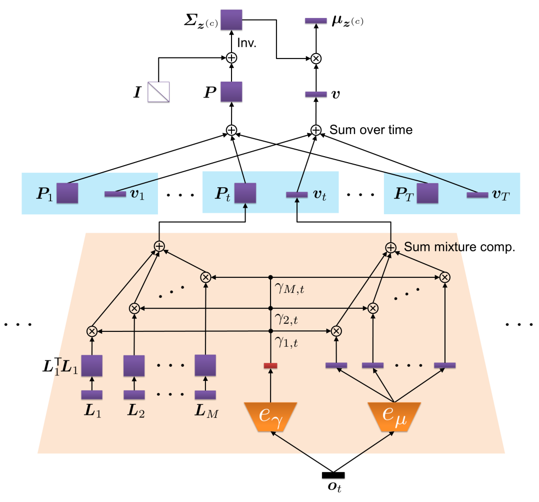

Fig. 3. Network architecture of sequence-level encoder. In experiments, the number of dimensions of $\boldsymbol{z}^{(c)}$ is greater than that of $\boldsymbol{o}_t$ $(R > D)$.

where the activation function for the output layer of $e_{\boldsymbol{\sigma}}$ is a sigmoid function.

IV. Experiments

A. Experimental setup

The task to evaluate the proposed model was speaker recognition using the TIMIT corpus [25]. There were a total 630 (192 female and 438 male) speakers and 10 short utterances per speaker. For model training, 9 out of 10 sentences per speaker were selected, and the remaining one sentence was kept for tests. The speech signals were sampled at a rate of 16 kHz and windowed by a 25-ms Hamming window with a 10-ms shift. Each frame was represented by 12-dimensional mel-frequency cepstral coefficients (MFCCs) without energy. Cepstral mean and variance normalization was performed. The MSR Identity Toolbox [26] was used for building GMM-UBM and i-vector based baseline systems. The gender-independent UBM consisted of 1024 Gaussian components with diagonal covariance matrices. The i-vectors were extracted by the total variability model with rank 200, and then they were modeled by the PLDA model consisting of 100 dimensions for the speaker subspace. The PLDA-scoring [3] with i-vector averaging was used. In the proposed model, standard feed-forward neural networks were used as $e_{\boldsymbol{\mu}}$, e_γ, $e_{\boldsymbol{\sigma}}$, f, g, and h. Full covariance was used in e and diagonal covariance was used in f and g. The network architectures were summarized in Table I. The weights were randomly initialized and were optimized using the RMSProp [27] algorithm with a learning rate of $1e-3$ and a dropout of 0.5 [28]. The batch size was 30. The hyperparameters in (16) were set to $\lambda_1 = 1e-2$, $\lambda_2 = 1e-3$, $\lambda_3 = 1e-4$. The L2 regularization [29] was applied to the weights of the sequence-level decoder h with the regularization strength $1e-6$. In test, the sequence-level latent variables were not sampled from Gaussian, i.e., $\boldsymbol{\mu}_{\boldsymbol{z}^{(c)}}$ instead of $\boldsymbol{z}^{(c)}$ was fed to h.

TABLE I
NETWORK ARCHITECTURES USED IN SPEAKER RECOGNITION ON THE TIMIT CORPUS.

Network	Number of units	Hidden/Output act.
$e_{\boldsymbol{\mu}}$	$12 \rightarrow 128 \rightarrow 100 \times 32$	LReLU/Linear
e_{γ}	$12 \rightarrow 256 \rightarrow 256 \rightarrow 32$	LReLU/Softmax
$e_{\boldsymbol{\sigma}}$	$12 \rightarrow 256 \rightarrow 100$	LReLU/Sigmoid
f	$12 \rightarrow 256 \rightarrow 6 \times 2$	ReLU/Linear,Softplus
g	$100 + 6 \rightarrow 256 \rightarrow 12 \times 2$	ReLU/Linear,Softplus
h	$100 \rightarrow 630$	NA/Softmax

TABLE II
SPEAKER RECOGNITION RATE ON THE TIMIT CORPUS.

	SRR [%]
GMM-UBM baseline	90.79
i-vector/PLDA baseline	90.95
Proposed model	**96.51**
w/o L2 regularization	89.84
w/o attention mechanism	95.24

B. Experimental results

Table II shows the speaker recognition rate (SRR). There was no significant difference between the performances of the GMM-UBM and i-vector based baseline systems. However, the proposed model outperformed the two baseline systems. This may be due to the nonlinear sequence-level feature extraction by neural networks. The L2 regularization greatly improved the performance of the proposed model. It can be said that avoiding the overfitting in the sequence-level decoder is important. The attention mechanism by $e_{\boldsymbol{\sigma}}$ further improved the performance of the proposed model. It seems that some of the observations that are not important for identifying speakers were discarded by the gate structure. A detailed investigation of the activations of $e_{\boldsymbol{\sigma}}$ is left to a later time.

V. CONCLUSIONS

A variational autoencoder based model was proposed for sequence modeling. The network architecture was inspired by the well-known i-vector/PLDA approach. Experiments on speaker recognition showed the effectiveness of the proposed model. Future work includes investigating the effectiveness of the proposed model in semi-supervised learning using the NIST SRE datasets and introducing multiple discriminators that identify phonemes, channels, and so on as well as speakers.

ACKNOWLEDGMENT

This work was partly supported by the Hoso Bunka Foundation.

REFERENCES

[1] N. Dehak, P. J. Kenny, R. Deahk, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," in *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[2] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," in *Proc. of Odyssey Speaker and Language Recognition Workshop*, 2010.

[3] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. of Interspeech 2011*, pp. 249–252, 2011.

[4] P. Rajan, A. Afanasyev, V. Hautamaki, and T Kinnunen, "From single to multiple enrollment i-vectors: practical PLDA scoring variants for speaker verification," in *Digital Signal Processing*, vol. 31, pp. 93–101, 2014.

[5] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," in *IEEE Trans. on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[6] E. Variani, X. Li, E. McDoermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. of ICASSP 2014*, pp. 4080–4084, 2014.

[7] Y. Chen *et al.*, "Locally-connected and convolutional neural networks for small footprint speaker recognition," in *Proc. of Interspeech 2015*, pp. 1136–1140, 2015.

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

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

[10] C. Li *et al.*, "Deep speaker: an end-to-end neural speaker embedding system," in *arXiv preprint*, arXiv:1705.02304, 2017.

[11] H. Bredin, "TristouNet: triplet loss for speaker turn embedding," in *Proc. of ICASSP 2017*, pp. 5430–5434, 2017.

[12] C. Zhang and K. Koishida, "End-to-end text-independent speaker verification with triplet loss on short utterances," in *Proc. of Interspeech 2017*, pp. 1487–1491, 2017.

[13] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: a unified embedding for face recognition and clustering," in *Proc. of CVPR 2015*, pp, 815–823, 2015.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. of the* 2nd *International Conference on Learning Representations*, 2014.

[15] M. Blaauw and J. Bonada, "Modeling and transforming speech using variational autoencoders," in *Proc. of Interspeech 2016*, pp. 1770–1774, 2016.

[16] H. Lee *et al.*, "Discriminative autoencoders for speaker verification," in *Proc. of ICASSP 2017*, pp. 5375–5379, 2017.

[17] J. Villalba, N. Brummer, and N. Dehak, "Tied variational autoencoder backends for i-vector speaker recognition," in *Proc. of Interspeech 2017*, pp. 1004–1008, 2017.

[18] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in *arXiv preprint*, arXiv:1406.5298, 2014.

[19] R. Caruana, "Multitask learning," in *Machine Learning*, vol. 28, pp. 41–75, 1997.

[20] K. Kazumi, Y. Nankaku, and K. Tokuda, "Factor analyzed voice models for HMM-based speech synthesis," in *Proc. of ICASSP 2010*, pp. 4234–4237, 2010.

[21] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. of EMNLP 2015*, pp. 1412–1421, 2015.

[22] A. Das, J. Li, R. Zhao, and Y. Gong, "Advancing connectionist temporal classification with attention modeling," in *Proc. of ICASSP 2018*, pp. 4769–4773, 2018.

[23] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. of EMNLP 2014*, pp. 1724–1734, 2014.

[24] A. van den Oord *et al.*, "Conditional image generation with PixelCNN decoders," in *arXiv preprint*, arXiv:1606.05328, 2016.

[25] J. S. Garofolo *et al.*, "DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM," 1993.

[26] S. O. Sadjadi, M. Slaney, and L. Heck, "MSR Identity Toolbox v1.0: A MATLAB toolbox for speaker recognition research," 2013.

[27] S. Ruder, "An overview of gradient descent optimization algorithms," in *arXiv preprint*, arXiv:1609.04747, 2016.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," in *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems 4*, pp. 950–957, 1992.