Unsupervised Mismatch Localization in Cross-Modal Sequential Data

Wei Wei[*], Huang Hengguan[*], Gu Xiangming
School of Computing
National University of Singapore
{weiwei, hengguan, xiangming}@comp.nus.edu.sg

Wang Hao
Department of Computer Science
Rutgers University
hoguewang@gmail.com

Wang Ye
School of Computing
National University of Singapore
wangye@comp.nus.edu.sg

Abstract

Content mismatch usually occurs when data from one modality is translated to another, e.g. language learners producing mispronunciations (errors in speech) when reading a sentence (target text) aloud. However, most existing alignment algorithms assume the content involved in the two modalities is perfectly matched and thus leading to difficulty in locating such mismatch between speech and text. In this work, we develop an unsupervised learning algorithm that can infer the relationship between content-mismatched cross-modal sequential data, especially for speech-text sequences. More specifically, we propose a hierarchical Bayesian deep learning model, named mismatch localization variational autoencoder (ML-VAE), that decomposes the generative process of the speech into hierarchically structured latent variables, indicating the relationship between the two modalities. Training such a model is very challenging due to the discrete latent variables with complex dependencies involved. We propose a novel and effective training procedure which estimates the hard assignments of the discrete latent variables over a specifically designed lattice and updates the parameters of neural networks alternatively. Our experimental results show that ML-VAE successfully locates the mismatch between text and speech, without the need for human annotations for model training.

1 Introduction

Sequential data is prevalent in daily life and usually comes in multiple modalities simultaneously, such as video with audio, speech with text, etc. Research problems about multi-modal sequential data processing have attracted great attention, especially on the alignment problem, such as aligning a video footage with actions [1] or poses [2], aligning speech with its text scripts [3], aligning audio with tags [4], etc. However, existing alignment methods all work based on an assumption that the content of data from multiple modalities should be matched. This results in difficulty in locating the mismatch between the two modalities.

The content mismatch could come from various aspects. For example, mispronounced words in speech or incorrect human annotation will cause mismatch between speech and text; actors not following scripts will cause mismatch between the video and the pre-scripted action list. Locating such content mismatch is an important task and has many potential applications. For example, locating the content mismatch between speech and text can help detect mispronunciations produced by the speaker, which is crucial for language learning.

To address this issue, we propose a hierarchical Bayesian deep learning model, dubbed mismatch localization variational autoencoder (ML-VAE), which is a hierarchical structured variational autoencoder (VAE) containing several hierarchically structured discrete latent variables. Our proposed model aims at performing content mismatch localization

[*]These authors contributed equally to this work.

between cross-modal sequential data, e.g. speech and the corresponding text. This is made possible via decomposing the generative process of speech into hierarchically structured latent variables, indicating the relationship between the two modalities.

We focus on applying our model to the mispronunciation localization task in which we propose to locate mispronounced phonemes in the speech produced by a speaker. Traditional speech-text alignment approaches, such as forced alignment [5] and connectionist temporal classification (CTC) segmentation [6], all work under the assumption that the speaker has correctly pronounced all the words, and thus they are not capable of detecting mispronunciation. At the same time, traditional mispronunciation detection methods mainly focus on training a phoneme classifier with labeled speech from second language (L2) speakers [7], which means they are not able to locate the mispronunciation in the speech. Besides, these methods need to be trained on a large number of human-annotated samples, whose annotation process is difficult, time-consuming and requires support from professional linguists. Unlike existing methods, our proposed ML-VAE does not need any human annotation during the training stage.

One challenge for ML-VAE is that training such an architecture is very difficult due to the discrete latent variables with complex dependencies. To address this challenge, we propose a novel and effective training procedure which estimates the hard assignments of the discrete latent variables and updates the neural network parameters alternatively.

The main contributions of our work include:

- We propose a hierarchical Bayesian deep learning model ML-VAE, to address the problem of content mismatch localization from cross-modal sequential data.
- To address the challenge of inferring the latent discrete variables with complex dependencies involved in ML-VAE, we propose a novel and effective alternating inference and learning procedure.
- We apply ML-VAE to the mispronunciation localization task and experiments are performed on a non-native English corpus to demonstrate its effectiveness in terms of unsupervisedly locating the mispronunciation segments in the speech.

2 Related Work

Cross-Modal Sequential Data Alignment. There have been several studies on aligning sequential data from different modalities. For example, for video-action alignment, most existing work focuses on learning spatio-temporal relations among the frames. [8] proposes a self-supervised learning approach named temporal cycle consistency learning to learn useful features for video action alignment. [9] proposes to learn a normalized human post feature to help perform the alignment task. [1] adopts an unsupervised way to align the actions in a video with its text description.

As for speech-text alignment, it is often referred to as the forced alignment (FA) task, where a phoneme sequence is aligned with its corresponding speech. The traditional method to solve this alignment task is to run the Viterbi decoding algorithm [10] on a Hidden-Markov-Model-based (HMM-based) acoustic model with a given text sequence to get the alignment result. There are several tools available to perform FA [5]. A more recent study by [6] proposes to use the CTC output to perform speech-text alignment. However, these existing studies assume perfect match in the content of the cross-modal sequential data, making it impossible to locate the possible content mismatch from data.

Variational Autoencoders. Variational autoencoder (VAE) [11] is proposed to learn the latent representations of real-world data by introducing latent variables. It adopts an encoder to approximate the posterior distribution of the latent variable and and an decoder to model the data distribution. VAEs have a wide range of applications, such as data generation [12], representation learning [13], etc. As an extension to VAE for handling sequential data, e.g. speech, variational recurrent neural network (VRNN) [14] introduces the latent variables into the hidden states of a recurrent neural network (RNN), allowing the complex temporal dependency across time steps to be captured. Along a similar line of research, [15] proposes the structured VAE, which integrated the conditional-random-field-like structured probability graphical model with VAEs to capture the latent structures of the video data. Factorized hierarchical variational autoencoder (FHVAE) [16] improves upon VRNN and SVAE through introducing two dependent latent variables at different time scales, which enables the learning of disentangled and interpretable representations from sequential data. However, the aforementioned hierarchical models are trained by directly optimizing the evidence lower bound (ELBO), which may fail when discrete latent variables with complex dependencies are involved. To address this issue, we propose a novel learning procedure to optimize our ML-VAE.

To deal with data from different modalities, [17] proposes a cross-modal VAE to capture both the intra-modal and cross-modal associations from input data. [18] proposes a VAE-based method to perform cross-modal alignment of latent spaces. However, existing work on processing cross-modal data mostly focuses on learning the relationship between modalities, while ignoring the potential mismatch between them. Therefore, we propose the ML-VAE to solve this issue by performing mismatch localization.

3 Background: Speech-Text Alignment

The speech-text alignment problem can be formulated as follows. Two sequences are given as inputs: 1) speech feature sequence $\mathcal{X} = (x_1, ..., x_T)$, and 2) phoneme sequence $\mathcal{C} = (c_1, ..., c_L)$. T represents the number of frames in the speech and L is the number of phonemes in the phoneme sequence; therefore $T > L$. The alignment result is an aligned phoneme sequence $\bar{\mathcal{C}} = (\bar{c}_1, ..., \bar{c}_T)$. $\bar{\mathcal{C}}$ can be treated as a repeated version of $\mathcal{C}$ with each element c_l repeating for d_l times, indicating that c_l lasts for d_l frames.

Existing work addresses the speech-text alignment problem by assuming that each phoneme is produced by following a single-path lattice representing a sequence of frames. Fig. 1 demonstrates the partial lattice for the l-th phoneme. From the initial state 0, there is only one path pointing to the state 1, standing for the l-th phoneme, c_l. Then at each time step, since each phoneme may last for several frames, it either still holds at state 1 (denoted by H), or moves forward to the final state 2 (denoted by S).

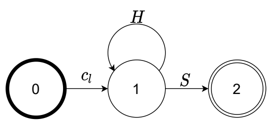

Figure 1: Single-path lattice.

The complete lattice can be constructed by connecting all the phoneme-level lattices together, according to the phoneme sequence $\mathcal{C}$. Then the aligned phoneme sequence $\bar{\mathcal{C}}$ is obtained by running dynamic programming (DP) algorithms. Taking the traditional HMM-based automatic speech recognition (ASR) models [19] as an example, the Viterbi decoding algorithm [10] is applied to search the most likely alignment result $\bar{\mathcal{C}}$.

4 Problem Formulation

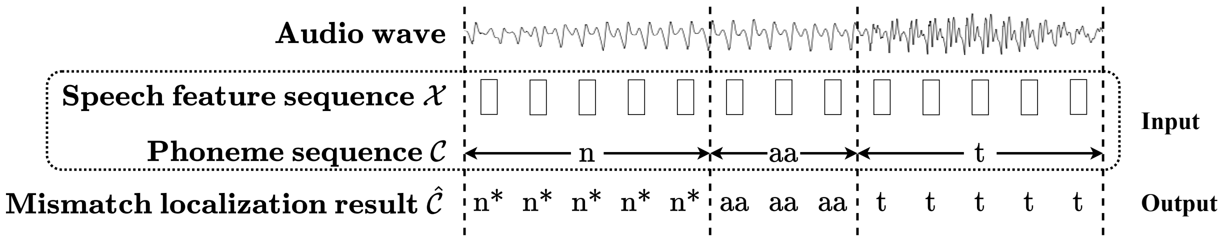
Audio wave
Speech feature sequence $\mathcal{X}$
Phoneme sequence $\mathcal{C}$
n
aa
t
Input
Mismatch localization result $\hat{\mathcal{C}}$
n* n* n* n* n* aa aa aa t t t t t
Output

Figure 2: Problem formulation.

Here, we propose to generalize the traditional speech-text alignment problem by considering the content mismatch between the input sequences (see Fig. 2). Specifically, given two cross-modal sequences: (1) a speech feature sequence $\mathcal{X} = (x_1, ..., x_T)$ as the source sequence, and (2) a phoneme sequence $\mathcal{C} = (c_1, ..., c_L)$ as the target sequence, our goal is to identify the mismatched target elements of $\mathcal{C}$ (i.e. mispronounced phonemes) while locating the corresponding elements in the source sequence (i.e. incorrect speech segments). Concretely, let $\mathcal{C}' = (c'_1, ..., c'_L)$ be the mismatch-identified target sequence; then $c'_l = c^*_l$ if c'_l is identified as mismatched content (i.e. a mispronounced phoneme), and $c'_l = c_l$ if c'_l is identified as matched content (i.e. a correctly pronounced phoneme). The final localization result $\hat{\mathcal{C}} = (\hat{c}_1, ..., \hat{c}_T)$ is therefore a repeated version of $\mathcal{C}'$ with each element c'_l repeating for d_l times, indicating that c'_l lasts for d_l frames.

5 Model

Overview. ML-VAE is a hierarchical Bayesian deep learning model [20] based on VAE [11]. Our model aims at performing content mismatch localization when aligning cross-modal sequential data. In this work, we focus on the speech-text mismatch localization task. This is made possible via decomposing the generative process of the speech into hierarchically structured latent variables, indicating the relationship between the two modalities.

ML-VAE is designed to use a hierarchical Gaussian mixture model (GMM) to model the match/mismatch between the sequences. Each Gaussian component corresponds to a type of mismatch, and the selection of the Gaussian component depends on the discrete latent variables of ML-VAE.

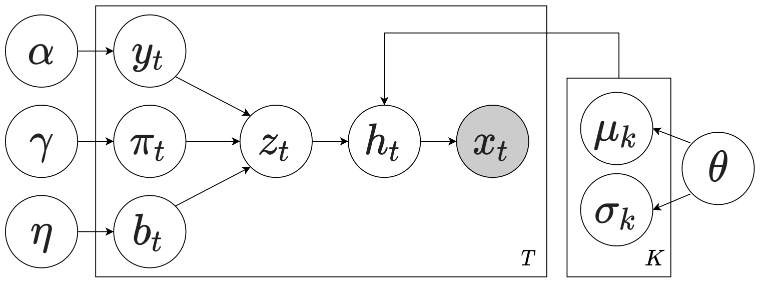

Figure 3: Graphical model for ML-VAE.

Latent Variables and Generative Process. Under the context of speech-text mismatch localization, since both the mismatch-identified phoneme sequence $\mathcal{C}'$ and the duration of each phoneme are unobservable, we introduce several latent variables to achieve the goal of mismatch localization for speech-text sequences. Instead of using a duration variable to describe the duration of each phoneme, we use a binary boundary variable sequence $\mathcal{B} = (b_1, ..., b_T)$, where $b_t = 1$ means the t-th frame marks the start of a phoneme segment, and thus $\sum_{t=1}^{T} b_t = L$. Besides, a binary correctness variable sequence $\Pi = (\pi_1, ..., \pi_T)$ is introduced to describe the matched/mismatched content in speech. Each entry $\pi_t \in \{0, 1\}$, with $\pi_t = 1$ if the t-th speech frame contains mismatched content (i.e. a mispronounced phoneme), and $\pi_t = 0$ otherwise.

To model the data generative process, at each time step t, we introduce three more latent variables: 1) y_t, which denotes the estimated phoneme, 2) z_t, which represents the Gaussian component indicator, and 3) h_t, which is the speech latent variable.

As shown in Fig. 3, we draw the estimated phoneme y_t from a categorical distribution. Meanwhile, the boundary variable b_t and the correctness variable π_t are assumed to be drawn from Bernoulli distributions. In the GMM part of ML-VAE, z_t, indicating the index of the selected Gaussian distribution, is drawn from a categorical distribution, and h_t, the speech latent variable, is drawn from the selected Gaussian distribution.

In the GMM part of ML-VAE, for each phoneme type, we use one component to model the matched content (i.e. correct pronunciation), and N_m components to model different mismatch types (i.e. mispronunciation variants). Such a design is motivated by the following observations: all correct pronunciations of the same phoneme are usually similar to each other, while its mispronunciations often severely deviate from the correct one and have multiple variants. Therefore, for N phoneme types, there are N components modeling correct pronunciations and $N * N_m$ components modeling mispronunciations, making the GMM part of ML-VAE contain a total number of $K = N + N * N_m$ components.

Given the design introduced above, the generative process of ML-VAE is as follows:

- For the t-th frame ($t = 1, 2, \dots, T$):
  - Draw the estimated phoneme $y_t \sim Categorical(\alpha)$.
  - Draw the boundary variable $b_t \sim Bernoulli(\gamma)$.
  - Draw the correctness variable $\pi_t \sim Bernoulli(\eta)$.
  - With y_t, b_t, and π_t, draw z_t from $z_t|y_t, \pi_t, b_t \sim Categorical(f_z(y_t, b_t, \pi_t))$, where $f_z(\cdot)$ denotes a learnable neural network.
  - Given that $z_t[k] = 1$, select the k-th Gaussian distribution and draw $h_t|z_t \sim \mathcal{N}(\mu_k, \sigma_k^2)$.
  - Draw $x_t|h_t \sim \mathcal{N}(f_\mu(h_k), f_\sigma(h_k))$, where $f_\mu(\cdot)$ and $f_\sigma(\cdot)$ denote learnable neural networks.

6 Learning

The learning of ML-VAE consists of two major components: latent variables $\Psi = \{\mathcal{Y}, \mathcal{B}, \Pi\}$ and neural network parameters $\Phi = \{\phi_p, \phi_b, \phi_h\}$, where $\mathcal{Y} = (y_1, ..., y_t)$, and ϕ_p, ϕ_b and ϕ_h denote parameters of the three modules of ML-VAE: phoneme estimator, boundary detector, and speech generator, respectively.

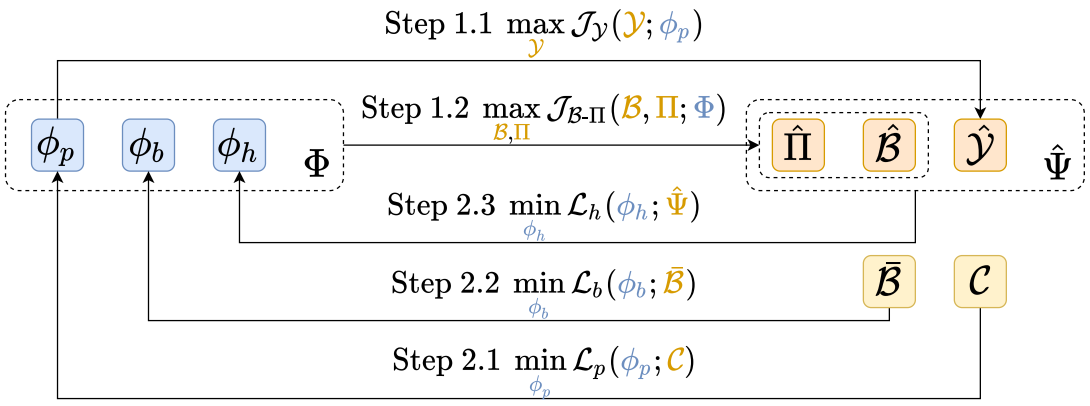
Step 1.1 $\max_{\mathcal{Y}} \mathcal{J}_{\mathcal{Y}}(\mathcal{Y}; \phi_p)$
Step 1.2 $\max_{\mathcal{B},\Pi} \mathcal{J}_{\mathcal{B}\text{-}\Pi}(\mathcal{B}, \Pi; \Phi)$
ϕ_p
ϕ_b
ϕ_h
Φ
$\hat{\Pi}$
$\hat{\mathcal{B}}$
$\hat{\mathcal{Y}}$
$\hat{\Psi}$
Step 2.3 $\min_{\phi_h} \mathcal{L}_h(\phi_h; \hat{\Psi})$
Step 2.2 $\min_{\phi_b} \mathcal{L}_b(\phi_b; \bar{\mathcal{B}})$
$\bar{\mathcal{B}}$
$\mathcal{C}$
Step 2.1 $\min_{\phi_p} \mathcal{L}_p(\phi_p; \mathcal{C})$

Figure 4: The training framework of ML-VAE.

Following the traditional variational inference and the training objective for FHVAE [16], the ELBO for joint training objective of ML-VAE can be written as:

$$\begin{aligned}\text{ELBO} = \sum_{t=1}^{T} \Big(& \mathbb{E}_{q(y_t, b_t, \pi_t | x)} \big[\log p(x_t | y_t, b_t, \pi_t) \big] \\ & - D_{\text{KL}}(q(y_t, b_t, \pi_t | x_t) || p(y_t, b_t, \pi_t) \Big),\end{aligned} \tag{1}$$

where D_{KL} is the function to calculate the Kullback–Leibler (KL) divergence, $q(y_t, b_t, \pi_t | x_t)$ is the joint approximate posterior distribution of the latent variables, $p(y_t, b_t, \pi_t)$ is the joint prior distribution.

However, unlike FHVAE, learning ML-VAE with such an objective function is very difficult due to ML-VAE's discrete latent variables with complex dependencies. Furthermore, we empirically tested an intuitive approach – using a hard expectation-maximization (EM) algorithm [21] – and found that the system would not reliably converge. As pointed by [22], lacking inductive bias for unsupervised training could render the model unidentifiable. We therefore improve upon this approach by providing pseudo labels to some latent variables, as introduced by [23]. Note that our learning algorithm remains unsupervised since it does not require any external data other than $\mathcal{X}$ and $\mathcal{C}$. Our new training procedure (see Fig. 4) can be described as:

- Step 1 (E step). Given the model parameters Φ, estimate the hard assignments of the latent variables $\hat{\Psi} = \{\hat{\mathcal{Y}}, \hat{\mathcal{B}}, \hat{\Pi}\}$ by:
  - Step 1.1. Given ϕ_p, estimate $\hat{\mathcal{Y}}$.
  - Step 1.2. Given Φ, estimate $\hat{\mathcal{B}}$ as well as $\hat{\Pi}$.
- Step 2 (M step). Given the hard assignments of the latent variables $\hat{\Psi}$, optimize the model parameters by:
  - Step 2.1. Given the phoneme sequence $\mathcal{C}$ as the training target, optimize ϕ_p .
  - Step 2.2. Given a forced alignment result $\bar{\mathcal{B}}$, optimize ϕ_b.
  - Step 2.3. Given the estimated hard assignments $\hat{\Psi}$ of latent variables, optimize ϕ_h.

Our approach differs from the standard hard EM algorithm in several aspects. First, since the phoneme sequence $\mathcal{C}$ is given under our problem setting, the phoneme estimator ϕ_p can be trained by directly optimizing the cross-entropy loss towards $\mathcal{C}$ (Step 2.1). Second, we can directly adopt the predictions of ϕ_p to obtain $\hat{\mathcal{Y}}$ (Step 1.1). Third, we design a lattice to assist approximating a maximum a posteriori (MAP) estimate of $\hat{\mathcal{B}}$ and $\hat{\Pi}$ (Step 1.2). Fourth, we find that directly using $\hat{\mathcal{B}}$ from Step 1.2 deteriorates the model performance; therefore in Step 2.2, we adopt a forced alignment result $\bar{\mathcal{B}}$ to train the boundary detector ϕ_b. It is worth noting that obtaining such an alignment in Step 2.2 does not require any external data other than $\mathcal{X}$ and $\mathcal{C}$.

6.1 Step 1: Estimation of the Hard Assignments $\hat{\Psi}$

Next, we discuss details on estimating the hard assignments.

Estimation of $\hat{\mathcal{Y}}$. We obtain the estimated phoneme sequence $\hat{\mathcal{Y}}$ with:

$$\hat{\mathcal{Y}} = \underset{\mathcal{Y}}{\operatorname{argmax}}\, \mathcal{J}_{\mathcal{Y}}(\mathcal{Y}; \phi_p) = \underset{\mathcal{Y}}{\operatorname{argmax}}\, q_{\phi_p}(\mathcal{Y}|\mathcal{X}), \tag{2}$$

where $q_{\phi_p}(\mathcal{Y}|\mathcal{X})$ is the variational distribution calculated by the phoneme estimator ϕ_p to approximate the intractable true posterior $p(\mathcal{Y}|\mathcal{X})$.

Estimation of $\hat{\mathcal{B}}$ and $\hat{\Pi}$. To estimate $\hat{\mathcal{B}}$ and $\hat{\Pi}$, we modify the lattice for the l-th phoneme introduced in Section 3 by introducing an extra path of content mismatch (see Fig. 5), with c_l or c_l^* representing the matched/mismatched content. For each path, it works similarly to the original lattice in Fig. 1.

With the help the phoneme sequence $\mathcal{C}$, we can straightforwardly build a sentence-level lattice by combining the corresponding phoneme-level lattices. Then $\hat{\mathcal{B}}$ and $\hat{\Pi}$ can be estimated by the following MAP estimator given the sentence-level lattice:

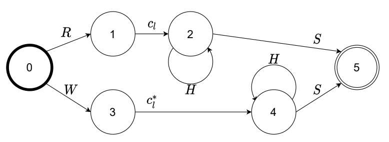

Figure 5: Proposed lattice.

$$\begin{aligned}\hat{\mathcal{B}}, \hat{\Pi} &= \underset{\mathcal{B},\Pi}{\operatorname{argmax}}\, \mathcal{J}_{\mathcal{B}\text{-}\Pi}(\mathcal{B}, \Pi; \Phi) \\ &= \underset{\mathcal{B},\Pi}{\operatorname{argmax}} \prod_{t=1}^{T} p(y_t|y_1, ..., y_{t-1}) \frac{q_{\phi_p}(y_t|x_t)}{p(y_t)},\end{aligned} \tag{3}$$

where $p(y_t)$ is the prior of y_t, which is usually estimated with the frequencies of different phonemes in the training dataset; $q_{\phi_p}(y_t|x_t)$ is the approximate posterior, which can be computed using the phoneme estimator. When calculating the transition probability $p(y_t|y_1, ..., y_{t-1})$ given the sentence-level lattice, three different cases are considered: (1) the consecutive frames belong to the same segment; (2) the consecutive two frames belong to different segments, and the current segment matches the phoneme sequence; (3) the consecutive two frames belong to different segments, and the current segment contains content mismatch (i.e. a mispronounced phoneme). Therefore, it can be defined as:

$$\begin{aligned}&p(y_t|y_1, ..., y_{t-1}) \\ &= \begin{cases} p(b_t = 0), & \text{if } y_t = y_{t-1} \\ p(b_t = 1)p(\pi_t = 0), & \text{if } y_t \neq y_{t-1} \text{ and } y_t \in \mathcal{C} \\ p(b_t = 1)p(\pi_t = 1), & \text{if } y_t \neq y_{t-1} \text{ and } y_t \in \mathcal{C}^*, \end{cases}\end{aligned} \tag{4}$$

where $\mathcal{C}^* = (c_1^*, ..., c_L^*)$ denotes the mismatched elements in the phoneme sequence $\mathcal{C}$ (i.e. mispronounced phonemes). $p(\pi_t = 0)$ and $p(\pi_t = 1)$ are approximated by $q_{\phi_h}(\pi_t = 0|x_t, y_t, b_t)$ and $q_{\phi_h}(\pi_t = 1|x_t, y_t, b_t)$, which can be calculated using the speech generator. Similarly, $p(b_t = 0)$ and $p(b_t = 1)$ are approximated by $q_{\phi_b}(b_t = 0|x_t)$ and $q_{\phi_b}(b_t = 1|x_t)$, which can be obtained from boundary detector. Then our MAP estimation can be achieved by using a classic DP algorithm, whereby $\hat{\mathcal{B}}$ and $\hat{\Pi}$ are backtracked along the optimal DP path. Meanwhile, the optimal DP path also yields the mismatch localization result $\hat{\mathcal{C}}$.

6.2 Step 2: Learning Model Parameters Φ

In this section, we present how ML-VAE's parameters $\Phi = \{\phi_p, \phi_b, \phi_h\}$ are learned given the estimated $\hat{\Psi}$ obtained from the previous stage. Implementation and parameterization details can be found in Appendix C.

Boundary Detector ϕ_b. The boundary detector takes the speech feature sequence $\mathcal{X}$ as input and outputs the probability $q_{\phi_b}(b_t|x_t)$. The boundary variable b_t is drawn from a Bernoulli distribution parameterized by a latent variable v_t, that is, $b_t \sim \text{Bernoulli}(v_t)$. Therefore here we models an auxiliary distribution $q_{\phi_b}(v_t|x_t)$. As introduced, we use the forced alignment result sequence $\bar{\mathcal{B}} = (\bar{b}_1, ..., \bar{b}_T)$ as the pseudo label to help the training process. The training objective is to minimize the loss $\mathcal{L}_b$:

$$\begin{aligned}\mathcal{L}_b(\phi_b; \bar{\mathcal{B}}) = -\sum_{t=1}^{T} \Big(&\mathbb{E}_{q_{\phi_b}(v_t|x_t)} \big[\log p(b_t = \bar{b}_t | v_t) \big] \\ &- \lambda_b D_{\mathrm{KL}}(q_{\phi_b}(v_t|x_t) || p(v_t)) \Big),\end{aligned} \tag{5}$$

where λ_b is a hyperparameter controlling the weight of the KL term, $q_{\phi_b}(v_t|x_t)$ is the approximate posterior distribution, and $p(v_t)$ is the prior distribution of v_t.

Phoneme Estimator ϕ_p. Similar to the boundary detector, the phoneme estimator takes the speech feature sequence $\mathcal{X}$ as input and outputs the probability $q_{\phi_p}(y_t|x_t)$. We use the pseudo label $\tilde{\mathcal{C}} = (\tilde{c}_1, ..., \tilde{c}_T)$ as our training target, which can be obtained by extending the phoneme sequence $\mathcal{C}$ according to the estimated duration of each phoneme in $\bar{\mathcal{B}}$. Therefore, the model is optimized by minimizing the negative log-likelihood:

$$\mathcal{L}_p(\phi_p; \mathcal{C}) = -\sum_{t=1}^{T} \log(q_{\phi_p}(y_t = \tilde{c}_t | x_t)). \tag{6}$$

Speech Generator ϕ_h. The speech generator ϕ_h aims to reconstruct the input speech feature sequence. It takes the speech feature sequence $\mathcal{X}$, along with the estimated latent variable values $\hat{\mathcal{Y}}$ and $\hat{\mathcal{B}}$ as input and reconstruct the speech feature sequence by following the generation process discussed in Section 5.

We use the variational inference to learn the speech generator ϕ_h and thus the ELBO loss is calculated by:

$$\begin{aligned}\mathcal{L}_r(\phi_h) = -\sum_{t=1}^{T} \Big(&\mathbb{E}_{q_{\phi_h}(h_t|x_t)} \big[\log p_{\phi_h}(x_t | h_t) \big] \\ &- \lambda_r D_{\mathrm{KL}}(q_{\phi_h}(h_t|x_t) || p(h_t)) \Big),\end{aligned} \tag{7}$$

where λ_r controls the weight of the KL term, $q_{\phi_h}(h_t|x_t)$ is the approximate posterior distribution, and $p(h_t)$ is the prior distribution of h_t.

Besides, we also augment the ELBO using the estimated correctness variable sequence $\hat{\Pi}$ obtained from Step 1 (E Step) as a supervision signal and the new loss is written as:

$$\mathcal{L}_h(\phi_h; \hat{\Psi}) = \mathcal{L}_r(\phi_h) + \lambda_l \mathcal{L}_l(\phi_h; \hat{\Psi}), \tag{8}$$

where λ_l is an importance weight; $\mathcal{L}_l(\phi_h; \hat{\Psi})$ is the negative log-likelihood loss, which is computed by:

$$\mathcal{L}_l(\phi_h; \hat{\Psi}) = -\sum_{t=1}^{T} \log q_{\phi_h}(\pi_t = \hat{\pi}_t | x_t, y_t, b_t), \tag{9}$$

where $q_{\phi_h}(\pi_t|x_t, y_t, b_t)$ is the approximate posterior distribution.

Reinforcement Learning. To better learn ML-VAE, we also use the reinforcement learning (RL) algorithm [24] to optimize Eq. 8. With the RL algorithm, ϕ_h is reinforced to sample the correctness variable sequence Π that produces larger $\mathcal{L}_h$. The RL objective is denoted as $\mathcal{L}_{rl}$, with details to be introduced in Appendix B. Therefore, $\mathcal{L}_h$ from Eq. 8 is replaced by $\mathcal{L}_{rl}$ in practice.

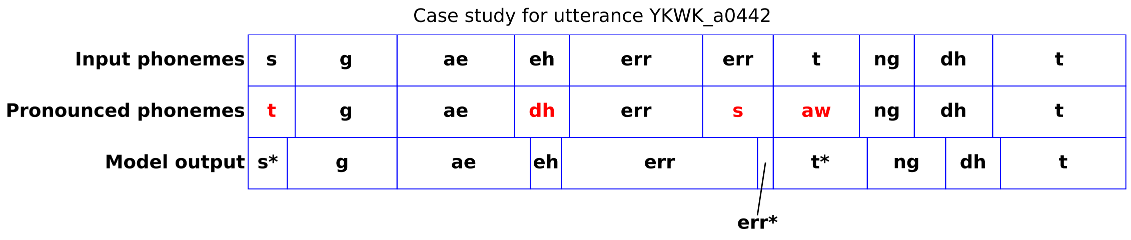

Figure 6: Case study for the last ten phonemes of the utterance YKWK_a0442. The first and second rows show the input phonemes and the actual phonemes pronounced by the speaker, respectively. The last row shows ML-VAE's predicted mispronunciation localization result.

7 Experiments

We evaluate our proposed ML-VAE's mismatch localization ability on the mispronunciation localization task. To perform experiments, we use the L2-ARCTIC speech corpus [25], which is a corpus containing speech of L2 speakers from different countries.

7.1 Evaluation Metrics

The F1 score is a commonly used metric to evaluate binary classification tasks, e.g., mispronunciation detection [7]. However, we find that F1 score is not suitable to evaluate the mispronunciation localization task, as it is computed without evaluating if the model successfully locates the detected mispronunciations in speech.

As such, we improve upon the traditional F1 score by proposing a set of new evaluation metrics. For each true positive (TP) case, we calculate the intersection over union (IoU) metric of its corresponding phoneme segment, which demonstrates the performance of localization, and then such IoU metrics of all TP cases are summed and denoted as $\mathrm{TP}_{\mathrm{ML}}$. After calculating $\mathrm{TP}_{\mathrm{ML}}$, the TP in equations to calculate the F1 score is replaced with $\mathrm{TP}_{\mathrm{ML}}$ to calculate our proposed metrics. For example, if there are two TP cases detected by the model, then instead of calculating F1 score with TP $= 2$, we calculate the IoU of these two cases' corresponding phoneme segment, e.g. 0.3 and 0.6 respectively. Then we calculate our proposed metrics with $\mathrm{TP}_{\mathrm{ML}} = 0.3 + 0.6$. Our proposed metrics are denoted as mismatch localization precision ($\mathrm{PR}_{\mathbf{ML}}$), recall ($\mathrm{RE}_{\mathbf{ML}}$), and F1 score ($\mathrm{F1}_{\mathbf{ML}}$). More details can be found in Appendix D.

7.2 Experimental Results

Baselines. To our knowledge, there is no existing work specifically deigned for mispronunciation localization. Therefore, we adapt some existing methods on related tasks (e.g. ASR) to perform mispronunciation localization, and compare our ML-VAE with them:

- **FA** [5], which performs forced alignment using a deep-neural-network-HMM-based (DNN-HMM-based) acoustic model.
- **Two-Pass-FA** [26], which first performs phoneme recognition based on a DNN-HMM-based acoustic model and then performs forced alignment on the recognized phoneme sequence and input speech.
- **w2v-CTC** [27], which is built with wav2vec 2.0 and trained with CTC loss; CTC-segmentation [6] is used to align the recognized phonemes with speech.

All models above are trained under the same problem setting, i.e. only the speech feature sequence $\mathcal{X}$ and phoneme sequence $\mathcal{C}$ which contains mismatch are used during the training process.

Table 1: Mispronunciation localization results.

Model	# Params	$\mathrm{PR}_{\mathbf{ML}}$	$\mathrm{RE}_{\mathbf{ML}}$	$\mathrm{F1}_{\mathbf{ML}}$
FA	3.3M	0.00	0.00	0.00
Two-Pass-FA	3.3M	0.64	0.96	0.77
w2v-CTC	352.5M	1.29	2.26	1.64
ML-VAE (ours)	25M	**8.20**	**13.57**	**10.22**

Mispronunciation Localization. As shown in Table 1, the vanilla forced alignment model (FA) fails to detect any mispronunciations due to its assumption that all the phonemes are correctly pronounced, and our ML-VAE significantly outperforms Two-Pass-FA in all three metrics.

Interestingly, though w2v-CTC is based on the state-of-the-art unsupervised ASR model wav2vec 2.0, which shows superior performance for ASR, it does not work well on the mispronunciation localization task; even with the largest number of parameters, it only slightly outperforms Two-Pass-FA, and underperforms our ML-VAE by a large margin. Our further analysis on w2v-CTC's output shows that such poor performance is mainly caused by the dataset; w2v-CTC is fine-tuned on the phoneme sequences with mispronunciations, and therefore its recognized phoneme sequence tends to contain mispronunciations as well. For example, if a speaker always mispronounces the phoneme 'b' as 'd', a w2v-CTC model trained with such data tends to predict 'd', instead of 'b' during inference. Another reason for w2v-CTC's poor performance is that the CTC-segmentation algorithm assumes correct pronunciations and therefore does not work well on speech-text sequences containing mispronunciations.

Table 2: Ablation study results.

Model	$PR_{\mathbf{ML}}$	$RE_{\mathbf{ML}}$	$F1_{\mathbf{ML}}$
ML-VAE	**8.20**	**13.57**	**10.22**
w/o reinforcement learning	6.46	11.97	8.39
w/ joint optimization	4.90	2.89	3.64
w/ $\hat{\mathcal{B}}$ & $\hat{\mathcal{Y}}$ estimated separately	7.31	10.85	8.73

Ablation Study. We present some ablation study results to demonstrate the effectiveness of our proposed learning algorithm. In Table 2, we first present the results of ML-VAE trained without RL (row 2), and ML-VAE trained with joint optimization method (row 3), as proposed by [16]. The mispronunciation localization results show that our proposed learning algorithm effectively improves upon the traditional joint optimization method. Besides, RL also helps to improve ML-VAE's mispronunciation localization performance.

We also present ablation study results to evaluate our estimation algorithm introduced in Section 6.2 (row 4), where $\hat{\mathcal{B}}$ and $\hat{\Pi}$ are jointly estimated using our proposed lattice. We compare our method with a traditional method, which estimates $\hat{\mathcal{B}}$ and $\hat{\Pi}$ separately based on a single-path lattice (e.g. Fig. 1). Details about this traditional algorithm can be found in Appendix E. The ablation study results show that using our proposed estimation algorithm, which is specifically designed for the mismatch localization task, yields better results.

Case Study. Besides quantitative analysis, we also provide a case study to showcase ML-VAE in Fig. 6. Each rectangle in Fig. 6 represents one phoneme segment. The first and second rows show the input phoneme sequence and the actual phonemes pronounced by the speaker, respectively. The last row shows ML-VAE's mispronunciation localization result. In the last row, a phoneme with a star sign (e.g. 't*') indicates that ML-VAE detects a mispronounced phoneme (e.g. a mispronounced 't'). It is shown that ML-VAE successfully detects three out of four mispronunciations with a reasonable localization result.

8 Conclusions

In this work, we present ML-VAE, a hierarchical Bayesian deep learning model to address the mismatch localization problem in cross-modal sequential data. We also propose a learning algorithm to optimize our proposed ML-VAE. We focus on applying ML-VAE to the speech-text mismatch localization problem and propose a set of new metrics to evaluate the model's mispronunciation localization performance. Our experimental results show that it can achieve superior performance than existing methods, including the adapted state-of-the-art unsupervised ASR model. We also perform ablation studies to verify the effectiveness of our training algorithm.

References

[1] Young Chol Song, Iftekhar Naim, Abdullah Al Mamun, Kaustubh Kulkarni, Parag Singla, Jiebo Luo, Daniel Gildea, and Henry A Kautz. Unsupervised Alignment of Actions in Video with Text Descriptions. In *IJCAI*, pages 2025–2031, 2016.

[2] Jogendra Nath Kundu, Ambareesh Revanur, Govind Vitthal Waghmare, Rahul Mysore Venkatesh, and R Venkatesh Babu. Unsupervised Cross-Modal Alignment for Multi-Person 3D Pose Estimation. In *ECCV*, 2020.

[3] Yu-An Chung, Wei-Hung Weng, Schrasing Tong, and James Glass. Unsupervised cross-modal alignment of speech and text embedding spaces. *arXiv preprint arXiv:1805.07467*, 2018.

[4] Xavier Favory, Konstantinos Drossos, Tuomas Virtanen, and Xavier Serra. Learning contextual tag embeddings for cross-modal alignment of audio and tags. In *ICASSP*, 2021.

[5] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. In *INTERSPEECH*, 2017.

[6] Ludwig Kürzinger, Dominik Winkelbauer, Lujun Li, Tobias Watzel, and Gerhard Rigoll. Ctc-segmentation of large corpora for german end-to-end speech recognition. In *SPECOM*, 2020.

[7] Wai-Kim Leung, Xunying Liu, and Helen Meng. CNN-RNN-CTC based end-to-end mispronunciation detection and diagnosis. In *ICASSP*, 2019.

[8] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *CVPR*, 2019.

[9] Jingyuan Liu, Mingyi Shi, Qifeng Chen, Hongbo Fu, and Chiew-Lan Tai. Normalized Human Pose Features for Human Action Video Alignment. In *ICCV*, 2021.

[10] G David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *ICML*, 2017.

[13] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

[14] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[15] Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. *Advances in neural information processing systems*, 29:2946–2954, 2016.

[16] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, pages 1878–1889, 2017.

[17] Dae Ung Jo, ByeongJu Lee, Jongwon Choi, Haanju Yoo, and Jin Young Choi. Cross-modal variational auto-encoder with distributed latent spaces and associators. *arXiv preprint arXiv:1905.12867*, 2019.

[18] Thomas Theodoridis, Theocharis Chatzis, Vassilios Solachidis, Kosmas Dimitropoulos, and Petros Daras. Cross-modal variational alignment of latent spaces. In *CVPR*, 2020.

[19] Biing Hwang Juang and Laurence R Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[20] Hao Wang and Dit-Yan Yeung. A survey on bayesian deep learning. *ACM Computing Surveys (CSUR)*, 53(5):1–37, 2020.

[21] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[22] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, 2019.

[23] Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *AISTATS*, 2020.

[24] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[25] Guanlong Zhao, Sinem Sonsaat, Alif O Silpachai, Ivana Lucic, Evgeny Chukharev-Khudilaynen, John Levis, and Ricardo Gutierrez-Osuna. L2-ARCTIC: A non-native English speech corpus. *Perception Sensing Instrumentation Lab*, 2018.

[26] Joseph Michael Tebelskis. *Speech recognition using neural networks*. Carnegie Mellon University, 1995.

[27] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*, 2020.

[28] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[29] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[30] Tao Lei, Yu Zhang, and Yoav Artzi. Training rnns as fast as cnns. 2018.

[31] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

A Overall Learning Algorithm

The overall algorithm to learn ML-VAE is shown in Algorithm 1.

Algorithm 1 Learning ML-VAE

```
Input: Speech feature sequence $\mathcal{X}$, phoneme sequence $\mathcal{C}$
Output: Mismatch localization result $\hat{\mathcal{C}}$
 1: Initialize the model parameters $\phi_p$, $\phi_b$, and $\phi_h$.
 2: Obtain the forced alignment result $\bar{\mathcal{B}}$.
 3: while not converged do
 4:     Estimate $\hat{\mathcal{B}}$ and $\hat{\Pi}$ with Eq. 3 and 4.
 5:     Using Eq. 6, optimize $\phi_p$ with the phoneme sequence $\mathcal{C}$.
 6:     Using Eq. 5, with the help of $\bar{\mathcal{B}}$, optimize $\phi_b$.
 7:     Given $\hat{\Pi}$, optimize $\phi_h$ using Eq. 8.
 8: end while
 9: Obtain the mismatch localization result $\hat{\mathcal{C}}$ with Eq. 3 and Eq. 4.
10: return $\hat{\mathcal{C}}$
```

B Reinforcement Learning

We use a reinforcement learning (RL) variant similar to [28] to train ML-VAE. We also follow [29] to introduce a reward term $\mathcal{R}(\Pi)$ as follows:

$$\mathcal{R}(\Pi) = -\mathcal{L}_h(\phi_h; \hat{\Psi}) - b(\mathcal{X}),$$

where $b(\mathcal{X})$ is a baseline term estimated by a neural network. This produces the following optimization objective to be minimized:

$$\begin{aligned}\mathcal{L}_{rl}(\phi_h; \hat{\Psi}) = \sum_{i=1}^{N_{mc}} \bigg(& \mathcal{L}_h(\phi_h; \hat{\Psi}) - H[\Pi] \\ & - \mathcal{R}(\Pi) * \log(\Pi = \Pi^{(i)} | \mathcal{X}, \mathcal{Y}, \mathcal{B}) + \mathcal{L}_{bl} \bigg),\end{aligned}$$

where $H[\pi]$ is the entropy of the distribution of the correctness variable, N_{mc} is the number of Monte Carlo samples, and $\Pi^{(i)}$ is the value of Π from the i-th sample. $\mathcal{L}_{bl}$ is the baseline loss term which is calculated by:

$$\mathcal{L}_{bl} = \text{MSE}(\mathcal{R}(\Pi), b(\mathcal{X})).$$

Therefore, $\mathcal{L}_h$ from Eq. 8 is replaced by $\mathcal{L}_{rl}$ in practice.

C Implementation Details

In this section, we introduce the implementation details of ML-VAE.

C.1 Boundary Detector

The boundary detector includes an encoder and a decoder. The encoder contains two LSTM layers, each with 512 nodes, followed by two fully connected (FC) layers with 128 nodes and ReLU activations, which are used to estimate the parameters (α and β) of the approximate posterior. The decoder has a fully connected layer, whose outputs are further transformed into a scalar with Softplus. During training, we set the weight of the KL term λ_b as 0.01.

C.2 Phoneme Estimator

The phoneme estimator contains two LSTM layers; each LSTM layer has 512 nodes, followed by two FC layers, each with 128 nodes and ReLU. They are followed by a Softmax layer to give the estimation of the phoneme.

C.3 Speech Generator

We adopt an encoder-decoder architecture to implement the speech generator. The encoder consists of three FC layers, each with 32 nodes, followed by four LSTM layers, each with 512 nodes. We use a FC layer with 512 nodes and ReLU to estimate the mean and variance of the Gaussian components. The decoder contains four bidirectional SRU layers [30], each with 512 nodes. They are followed by two FC layers, each with 120 nodes, to estimate the mean and variance of the data distribution. During training, λ_r and λ_l are set as 1 and 0.001 respectively.

Gaussian Component Selection. As described in the generative process, the Gaussian component indicator z_t is sampled from from $z_t|y_t, \pi_t, b_t \sim Categorical(f_z(y_t, b_t, \pi_t))$. In this section we describe the implementation of $f_z(\cdot)$, which is defined as:

$$f_z(y_t, b_t, \pi_t) = \text{softmax}(\rho_t * \epsilon + \delta_t), \tag{10}$$

where ρ_t is a scalar, which is estimated by a two-layer multilayer perceptron (MLP) taking as inputs y_t, b_t, and π_t. Each layer of the MLP contains 128 nodes and Sigmoid; ϵ is a small constant, which is set as 1×10^{-6} in our experiments; δ_t is an one-hot variable, i.e. $\delta_t[s] = 1$, and s is computed by:

$$s = \begin{cases} (j-1) * (N_m + 1) + 1, & \text{if } \pi_t = 0 \\ (j-1) * (N_m + 1) + 1 + k, & \text{if } \pi_t = 1, \end{cases} \tag{11}$$

where j denotes the phoneme label of y_t, i.e. $y_t[j] = 1$. In case that y_t is mispronounced, $k \in [1, N_m]$ denotes the k-th mispronunciation variant of the phoneme y_t, which is implemented by a Gumbel Softmax function [31], i.e. $\tau_t = \text{Gumbel}(\text{NN}(x_t, y_t))$, where $\tau_t[k] = 1$ and NN is a simple neural network with three 128-node FC layers .

D Proposed Evaluation Metrics

As introduced in the main paper, we propose a set of new metrics to evaluate the mode's performance on the mispronunciation localization task: mismatch localization precision ($\text{PR}_{\textbf{ML}}$), recall ($\text{RE}_{\textbf{ML}}$), and F1 score ($\text{F1}_{\textbf{ML}}$)..

Table 3: The confusion matrix. C stands for correct pronunciation. M stands for mispronunciation.

		Model output	
		C	M
Ground truth	C	TN	FP
	M	FN	TP

We first follow the confusion matrix (see Table 3) for mispronunciation detection to calculate the the intermediate metrics: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). (Note that in some studies, the TP in the confusion matrix is named as true negative, or true rejection.) Then as introduced in the main papaer, for all the TP cases detected by the model, we take the sum of their corresponding segments' intersection over union (IoU), which is denoted as TP_{IoU}. Then the final metrics are computed as:

$$\begin{aligned} \text{PR}_{\textbf{ML}} &= \frac{\text{TP}_{\text{IoU}}}{\text{TP} + \text{FP}} \\ \text{RE}_{\textbf{ML}} &= \frac{\text{TP}_{\text{IoU}}}{\text{TP} + \text{FN}} \\ \text{F1}_{\textbf{ML}} &= \frac{2 \times \text{PR}_{\textbf{ML}} \times \text{RE}_{\textbf{ML}}}{\text{PR}_{\textbf{ML}} + \text{RE}_{\textbf{ML}}} \end{aligned} \tag{12}$$

E Estimating $\hat{\mathcal{B}}$ and $\hat{\Pi}$ separately

In this section, we describe the algorithm which estimates the hard assignments for $\hat{\mathcal{B}}$ and $\hat{\Pi}$ separately. This algorithm is used to perform ablation study.

E.1 Estimation of $\hat{\mathcal{B}}$

Being different from our joint estimation algorithm discussed in the main paper, the boundary variable $\hat{\mathcal{B}}$ is first estimated without considering the pronunciation correctness.

When estimating $\hat{\mathcal{B}}$, we adopt the single-path lattice introduced in Section 3 of our main paper (see Fig. 1). Based on such a lattice, the following MAP estimator is used to estimate $\hat{\mathcal{B}}$:

$$\hat{\mathcal{B}} = \underset{\mathcal{B}}{\operatorname{argmax}} \prod_{t=1}^{T} p(y_t|y_1, ..., y_{t-1}) \frac{q_{\phi_p}(y_t|x_t)}{p(y_t)}. \tag{13}$$

The transition probability is calculated by considering if two consecutive frames belong to the same phoneme segment:

$$\begin{aligned} &p(y_t|y_1, ..., y_{t-1}) \\ &= \begin{cases} p(b_t = 0), & \text{if } y_t = y_{t-1} \\ p(b_t = 1), & \text{if } y_t \neq y_{t-1}, \end{cases} \end{aligned} \tag{14}$$

Similar to our joint estimation algorithm, this MAP estimation can be achieved by a classic dynamic programming (DP) algorithm. Then $\hat{\mathcal{B}}$ is obtained by backtracking along the optimal DP path.

E.2 Estimation of $\hat{\Pi}$

We then estimate $\hat{\Pi}$ given the $\hat{\mathcal{B}}$ obtained in the above procedure. If we are given the l-th phoneme segment which starts at u-th frame and ends at the v-th frame, we assume that all frames within this segment share the same pronunciation correctness label, that is, $\pi_s = \pi_{s+1} = \cdots = \pi_v = \pi'_l$, where π'_l denotes the pronunciation correctness of the l-th phoneme.

Therefore, we can obtain the MAP estimation of π'_l per segment:

$$\begin{aligned} \hat{\pi}'_l &= \underset{\pi'_l}{\operatorname{argmax}} \prod_{t=u}^{v} p(y_t|y_1, ..., y_{t-1}) \frac{q_{\phi_p}(y_t|x_t)}{p(y_t)} \\ &= \underset{\pi'_l}{\operatorname{argmax}} \begin{cases} p(\pi_s = 0) \prod_{t=u}^{v} \frac{q_{\phi_p}(y_t = c_l|x_t)}{p(y_t = c_l)}, & \text{if } \pi'_l = 0 \\ p(\pi_s = 1) \prod_{t=u}^{v} \frac{q_{\phi_p}(y_t \neq c_l|x_t)}{p(y_t \neq c_l)}, & \text{if } \pi'_l = 1, \end{cases} \end{aligned} \tag{15}$$

where $q_{\phi_p}(y_t|x_t)$, $q_{\phi_p}(y_t = c_l|x_t)$, and $q_{\phi_p}(y_t \neq c_l|x_t)$ are the approximate posteriors. $y_t = c_l$ denotes that the t-th frame is correctly pronounced ($\pi'_l = 0$). $y_t \neq c_l$ denotes that the t-th frame is mispronounced ($\pi'_l = 1$), that is, $q_{\phi_p}(y_t \neq c_l|x_t) = 1 - q_{\phi_p}(y_t = c_l|x_t)$, and $p(y_t \neq c_l) = 1 - p(y_t = c_l)$.