Unsupervised Mismatch Localization in Cross-Modal Sequential Data

Anonymous authors
Paper under double-blind review

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 that the content involved in the two modalities is perfectly matched, 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, dubbed mismatch localization variational autoencoder (ML-VAE), which 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. To address this challenge, we propose a novel and effective training procedure that alternates between estimating the hard assignments of the discrete latent variables over a specifically designed mismatch localization finite-state acceptor (ML-FSA) and updating the parameters of neural networks. 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 (Song et al., 2016) or poses (Kundu et al., 2020), aligning speech with its text scripts (Chung et al., 2018), aligning audio with tags (Favory et al., 2021), etc.

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.

In this work, we focus on the problem of mismatch localization between speech and text inputs. In other words, i.e., to locate the mispronunciations in the speech produced by a speaker. Most existing cross-modal alignment algorithms assume that data from a variety of modalities are matched to each other (Song et al., 2016; Kundu et al., 2020; Chung et al., 2018; Favory et al., 2021); this strong assumption leads to difficulty in locating the mismatch between the two modalities. In particular, recent speech-text alignment approaches (Kürzinger et al., 2020; McAuliffe et al., 2017) mostly work under the assumption that the speaker has correctly pronounced all the words and are therefore incapable of detecting mispronunciation. Although earlier studies Finke & Waibel (1997); Hazen (2006); Braunschweiler et al. (2010); Bell & Renals (2015) have considered the mismatch between speech and text, they require human-annotated speech to train an acoustic model. However, labeling such data is labor intensive and expensive. Similarly, traditional mispronunciation detection methods (Leung et al., 2019) also need to be trained on a large number of human-annotated speech samples from second language (L2) speakers, whose annotation process is even more time-consuming and requires professional linguists' support. Furthermore, these studies on mispronunciation detection can only detect which phonemes/words in the text input are mispronounced, without locating them in the speech.

To address these issues, we propose a hierarchical Bayesian deep learning model, dubbed mismatch localization variational autoencoder (ML-VAE), which aims at performing content mismatch localization between cross-modal sequential data without requiring any human annotation during the training stage. Our model is a hierarchically structured variational autoencoder (VAE) containing several hierarchically structured discrete latent variables. These latent variables describe the generative process of speech from L2 speakers and indicate the relationship between the two modalities.

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 that alternates between estimating the hard assignments of the discrete latent variables over a specifically designed finite-state automaton (FSA) and updating the parameters of neural networks.

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.
- Our ML-VAE is the first method that bridges finite-state automata and variational autoencoders; this is achieved via our proposed mismatch localization finite-state acceptor (ML-FSA), which allows the ML-VAE to locate the mismatch by searching for the best path in ML-FSA.
- 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; experiments on a non-native English corpus to demonstrate ML-VAE's 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. Dwibedi et al. (2019) propose a self-supervised learning approach named temporal cycle consistency learning to learn useful features for video action alignment. Liu et al. (2021) propose to learn a normalized human pose feature to help perform the alignment task. Song et al. (2016) adopt an unsupervised way to align the actions in a video with its text description. In terms of speech-text alignment, a recent study by Kürzinger et al. (2020) proposes using the CTC output to perform speech-text alignment. However, these aforementioned studies assume a perfect match in the content of the cross-modal sequential data, making it impossible to locate the content mismatch from the data.

Speech-text alignment is usually referred to as the forced alignment (FA) task (McAuliffe et al., 2017) in the speech processing community. There is a fairly long history of research related to FA. Traditional method is to run the Viterbi decoding algorithm (Forney, 1973) on a Hidden-Markov-Model-based (HMM-based) acoustic model with a given text sequence. Even though several FA studies have considered the particular problem setting where the text is not perfectly matched to the speech, they require training an acoustic model using a large amount of human-annotated speech, which is laborious and costly. For instance, Finke & Waibel (1997); Moreno et al. (1998); Moreno & Alberti (2009); Bell & Renals (2015); Stan et al. (2012) require an acoustic model for text-to-speech alignment with a modified lattice, and Bordel et al. (2012) requires an acoustic model to obtain the recognized phoneme sequence. Therefore, these methods fail to handle our problem setting, where the speech data from L2 speakers is unlabelled.

Variational Autoencoders Variational autoencoder (VAE) (Kingma & Welling, 2013) 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 (Mescheder et al., 2017), representation learning (Oord et al., 2017), etc. As an extension to VAE to handle sequential data (e.g., speech), the variational recurrent neural network (VRNN) (Chung et al., 2015) introduces 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, Johnson et al. (2016) propose the structured VAE, which integrates the conditional random field-like structured probability graphical model with VAEs to capture the latent structures of video data. Factorized hierarchical variational autoencoder (FHVAE) (Hsu et al., 2017) 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 hierarchical models mentioned above 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, Jo et al. (2019) propose a cross-modal VAE to capture both intra-modal and cross-modal associations from input data. Theodoridis et al. (2020) propose a VAE-based method to perform cross-modal alignment of latent spaces. However, existing work on processing cross-modal data focuses mainly 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 Problem Formulation

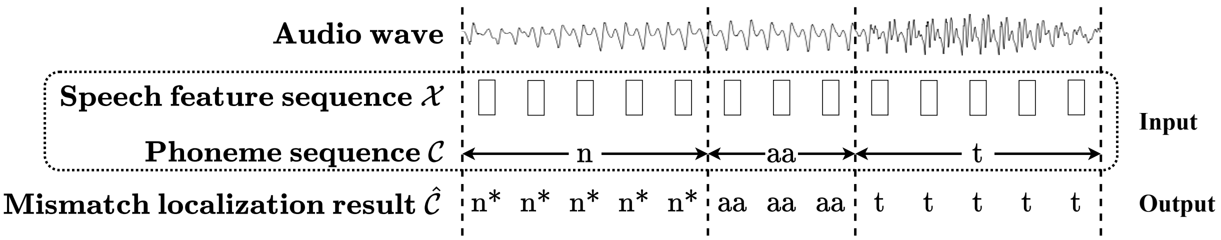
Audio wave
Speech feature sequence $\mathcal{X}$
Phoneme sequence $\mathcal{C}$
n
aa
t
Input
Mismatch localization result $\hat{\mathcal{C}}$
n* n* n* n* n* aa aa aa t t t t t
Output

Figure 1: Problem formulation.

Here, we propose to generalize the traditional speech-text alignment problem by considering the content mismatch between the input sequences (see Fig. 1). 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. We use the notation c^*_l to present mismatched content; 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.

4 Model

ML-VAE is a hierarchical Bayesian deep learning model (Wang & Yeung, 2020) based on VAE (Kingma & Welling, 2013). 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 by 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.

Latent Variables and Generative Process In 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.

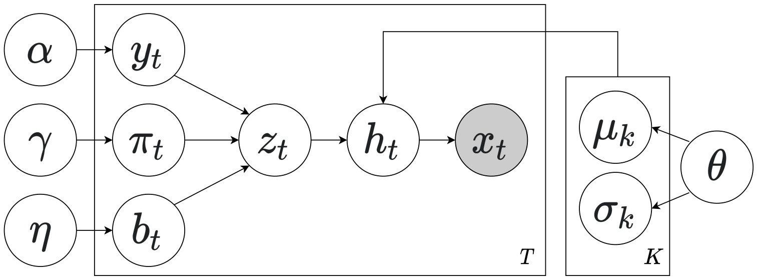

Figure 2: Graphical model for ML-VAE.

As shown in Fig. 2, 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 latent variable of speech, 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 ($t = 1, 2, \ldots, T$):
  - 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.

An Example We will take an audio recording of reading the word 'NOT' as an example to explain the generative process. More specifically, the phoneme sequence contains three phonemes: **n**, **aa**, and **t**. The estimated phoneme y_t is sampled to estimate the phoneme pronounced by the speaker at each time step. The boundary variable b_t is sampled to determine whether the speaker starts to pronounce the next phoneme at time step t (e.g., from **n** to **aa**). The correctness variable π_t is then sampled to determine whether this phoneme (e.g., **aa**) is correctly pronounced (e.g., whether **aa** is mispronounced as another phoneme). Based on y_t, b_t, and π_t, if the phoneme (e.g., **aa**) is correctly pronounced, the Gaussian component indicator z_t will select the Gaussian distribution for the correct pronunciation to generate the speech latent variable h_t; otherwise, based on how the phoneme is mispronounced (e.g, **aa** mispronounced as **ae**), z_t will select the corresponding Gaussian distribution to generate the speech latent variable h_t.

5 Mismatch Localization Finite-State Acceptor

In this section, we describe our proposed mismatch localization finite-state acceptor (ML-FSA) that bridges finite-state automata and variational autoencoders, allowing our ML-VAE to locate the mispronounced segment in the speech by efficiently searching the best path in ML-FSA. Our ML-FSA is a special type of finite-state acceptor (FSA) that describes possible hypothesis of mispronunciations and phoneme boundaries. We show ML-FSA for the l-th phoneme, c_l, in the given phoneme sequence in Fig. 3. Specifically, from the initial state (state 0), it can transit to state 1 or 3 based on pronunciation correctness (R stands for correct pronunciation and W stands for mispronunciation). This further leads to two different paths, one for correct pronunciation (denoted by c_l), and the other one for mispronunciation (denoted by c_l^*). In each path, at each time step, since each phoneme may last for several frames, it either still holds at the current state (denoted by H), or moves forward to the final state 5 (denoted by S). With the help of the phoneme sequence $\mathcal{C}$, we can naturally build a sentence-level ML-FSA by combining the corresponding ML-FSA for each phoneme.

In contrast to weighted finite state transducers (WFSTs) Mohri et al. (2002), which only model the combination of acoustic units (e.g., phonemes), our ML-FSA models the correctness of pronunciation (R and W), the phoneme type (c_l, c_l^*), and the phoneme boundary (H and S), which can jointly detect and segment the mispronunciation in the speech and thus perform *unsupervised content mismatch localization.* Furthermore, WFSTs require training a supervised acoustic model for weighting transitions between states, while our ML-FSA adopts the unsupervised ML-VAE to calculate all the weights of transitions. The details on locating the mispronunciation using ML-FSA are described in the next section. Note that ML-FSA takes all possible mispronunciation variants as a single symbol (c_l^*), which trades fidelity for efficiency. This is advantageous for addressing our problem settings: (1) our task focuses on distinguishing between the correct pronunciation and mispronunciation, rather than identifying the mispronunciation variants; (2) combining mispronunciation variants for each phoneme can dramatically reduce the computational cost of the dynamic programming (DP) algorithm searching for the optimal path.

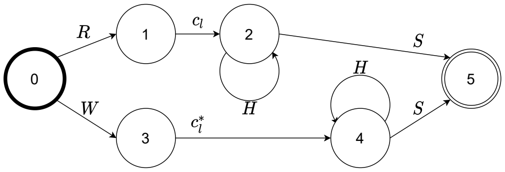

Figure 3: Proposed ML-FSA for the l-th phoneme, c_l.

6 Learning

The learning of ML-VAE consists of two main 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.

Following the traditional variational inference and the training objective for FHVAE (Hsu et al., 2017), the ELBO for the joint training objective of ML-VAE can be written as:

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

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 (Moon, 1996) – and found that the system would not reliably converge. As pointed by Locatello et al. (2019), lacking inductive bias for unsupervised training could render the model unidentifiable. Therefore, we improve this approach by providing pseudo-labels to some latent variables, as introduced by Khemakhem et al. (2020). 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:

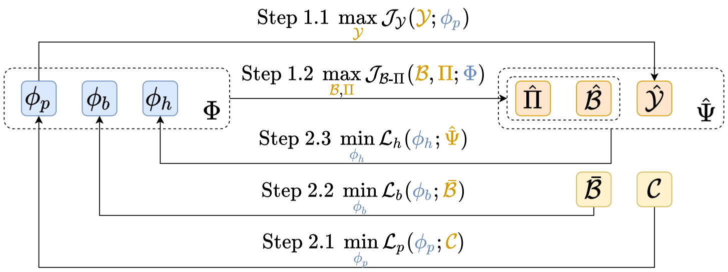
Step 1.1 $\max_{\mathcal{Y}} \mathcal{J}_{\mathcal{Y}}(\mathcal{Y}; \phi_p)$
Step 1.2 $\max_{\mathcal{B},\Pi} \mathcal{J}_{\mathcal{B}\text{-}\Pi}(\mathcal{B}, \Pi; \Phi)$
ϕ_p
ϕ_b
ϕ_h
Φ
$\hat{\Pi}$
$\hat{\mathcal{B}}$
$\hat{\mathcal{Y}}$
$\hat{\Psi}$
Step 2.3 $\min_{\phi_h} \mathcal{L}_h(\phi_h; \hat{\Psi})$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

- 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 $\hat{\mathcal{B}}$ and $\hat{\Pi}$.
- 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 an FSA 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 the details of 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 $\hat{\Pi}$ via ML-FSA Given the sentence-level ML-FSA introduced in Sec. 5, $\hat{\mathcal{B}}$ and $\hat{\Pi}$ can be estimated by the following MAP estimator:

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

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 FSA, 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:

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

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 the 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 Sec. 8.

6.2.1 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 aid the training process. The training objective is to minimize the loss $\mathcal{L}_b$:

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

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.

6.2.2 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 $\hat{\mathcal{B}}$. Therefore, the model is optimized by minimizing the negative log-likelihood:

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

6.2.3 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 values of the latent variables $\hat{\mathcal{Y}}$ and $\hat{\mathcal{B}}$ as input and reconstructs the speech feature sequence following the generation process discussed in Section 4.

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

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

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.

6.3 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 $\hat{\mathcal{C}}$
Initialize the model parameters $\phi_p$, $\phi_b$, and $\phi_h$.
Obtain the forced alignment result $\bar{\mathcal{B}}$.
while not converged do
    Estimate $\hat{\mathcal{B}}$ and $\hat{\Pi}$ with Eq. 3 and 4.
    Using Eq. 6, optimize $\phi_p$ with the phoneme sequence $\mathcal{C}$.
    Using Eq. 5, with the help of $\bar{\mathcal{B}}$, optimize $\phi_b$.
    Given $\hat{\Pi}$, optimize $\phi_h$ using Eq. 8.
end while
Obtain the mismatch localization result $\hat{\mathcal{C}}$ with Eq. 3 and Eq. 4.
return $\hat{\mathcal{C}}$
```

7 ML-VAE with Reinforcement Learning

We further propose a variant of ML-VAE by using the reinforcement learning (RL) algorithm (Sutton & Barto, 2018) to better reason the distribution of the correctness variable during training, which is named as

ML-VAE-RL. We use an RL variant similar to that proposed by Williams (1992). We also follow the work by Xu et al. (2015) to introduce a reward term $\mathcal{R}(\Pi) = -\mathcal{L}_h(\phi_h; \hat{\Psi}) - b(\mathcal{X})$, where $b(\mathcal{X})$ is a baseline term estimated by a fully-connected neural network. Then the loss in Eq. 8 can be re-written as:

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

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 i-th sample drawn from the distribution of the correctness variable. $\mathcal{L}_{bl}$ is the baseline loss term which is calculated by $\mathcal{L}_{bl} = \text{MSE}(\mathcal{R}(\Pi), b(\mathcal{X}))$.

With the RL algorithm, ϕ_h is reinforced to sample the correctness variable sequence Π that produces larger $\mathcal{L}_h(\phi_h; \hat{\Psi})$.

8 Implementation Details

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

8.1 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.

8.2 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.

8.3 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 (Lei et al., 2018), 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 to 1 and 0.001, respectively.

Gaussian Component Selection As described in the generative process, the Gaussian component indicator z_t is sampled 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 $f_z(y_t, b_t, \pi_t) = \text{softmax}(\rho_t * \epsilon + \delta_t)$, where ρ_t is a scalar 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 a Sigmoid activation function; 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:

$$s = \begin{cases} (j-1) * (N_m + 1) + 1, & \text{if } \pi_t = 0 \\ (j-1) * (N_m + 1) + 1 + k, & \text{if } \pi_t = 1, \end{cases} \tag{10}$$

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

(Jang et al., 2016), 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 .

9 Experiments

We evaluate our proposed ML-VAE and ML-VAE-RL on the mispronunciation localization task to test their mismatch localization ability. We first conduct experiments on a synthetic dataset (Mismatch-AudioMNIST) and then further apply our proposed models to a real-world speech-text dataset (L2-ARCTIC).

9.1 Evaluation Metrics

The F1 score is a commonly used metric to evaluate binary classification tasks, e.g., mispronunciation detection (Leung et al., 2019). 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 TP_{ML}. After calculating TP_{ML}, the TP in equations to calculate the F1 score is replaced with TP_{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 $\text{TP}_{\text{ML}} = 0.3 + 0.6$. Our proposed metrics are denoted as mismatch localization precision ($\text{PR}_{\mathbf{ML}}$), recall ($\text{RE}_{\mathbf{ML}}$), and F1 score ($\text{F1}_{\mathbf{ML}}$).

To be more specific, we first calculate the the intermediate metrics: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Then 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: $\text{PR}_{\mathbf{ML}} = \frac{\text{TP}_{\text{IoU}}}{\text{TP}+\text{FP}}$, $\text{RE}_{\mathbf{ML}} = \frac{\text{TP}_{\text{IoU}}}{\text{TP}+\text{FN}}$, $\text{F1}_{\mathbf{ML}} = \frac{2\times\text{PR}_{\mathbf{ML}}\times\text{RE}_{\mathbf{ML}}}{\text{PR}_{\mathbf{ML}}+\text{RE}_{\mathbf{ML}}}$

9.2 Baselines

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

- **FA** (McAuliffe et al., 2017), which performs forced alignment using a deep-neural-network-HMM-based (DNN-HMM-based) acoustic model.
- **Two-Pass-FA** (Tebelskis, 1995), 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** (Baevski et al., 2020), which is built with wav2vec 2.0 and trained with CTC loss; CTC-segmentation (Kürzinger et al., 2020) 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.

9.3 Synthetic Dataset: Mismatch-AudioMNIST

We first test our proposed models on a synthetic dataset, named as **Mismatch-AudioMNIST**, which is built based on the AudioMNIST (Becker et al., 2018) dataset. Our synthetic dataset contains 3000 audio samples, each produced by concatenating three to seven spoken digits randomly selected from the original

AudioMNIST. To simulate the content mismatch between audio and the corresponding text annotations, we randomly select 20.1% of the spoken digits as mismatched content, thereby labeling them as random digits. The total duration of Mismatch-AudioMNIST is 2.67 hours. The dataset is split into training, validation, and test sets by a 60:20:20 ratio, and the durations of the three sets are 96.3, 33.2, and 30.6 minutes respectively.

Mispronunciation Localization Results Table 1 shows the mispronunciation localization results of ML-VAE and ML-VAE-RL on the synthetic dataset Mismatch-AudioMNIST. We can see that all three baselines perform poorly for our mismatch localization task. More specifically, the vanilla forced alignment model (FA) fails to locate any mispronunciations due to its assumption that all the digits are correctly pronounced. Both of our proposed models (ML-VAE and ML-VAE-RL) are much superior to the baseline models, Two-Pass-FA and w2v-CTC, demonstrating the effectiveness of the proposed method in handling the unsupervised mismatch localization task. Note that in our problem settings, our training data contains mislabeled spoken digits and such mislabelling is unknown. This poses great challenges for acoustic model training. By further looking into the recognition results of Two-Pass-FA and w2v-CTC, we find that their recognition results from the acoustic model are very poor in terms of the mislabeled spoken digits, consequently limiting the baseline models' ability to detect such mislabels in the second stage processing.

Table 1: Mispronunciation localization results on Mismatch-AudioMNIST.

Model	# Params	PR$_{\textbf{ML}}$%	RE$_{\textbf{ML}}$%	F1$_{\textbf{ML}}$%
FA	3.3M	0.00	0.00	0.00
Two-Pass-FA	3.3M	6.22	2.43	2.28
w2v-CTC	352.5M	1.72	3.84	2.30
ML-VAE (ours)	24M	28.42	27.60	27.67
ML-VAE-RL (ours)	25M	**30.67**	**30.32**	**30.28**

9.4 Real-World Dataset: L2-ARCTIC

We further apply ML-VAE and ML-VAE-RL to a real-world dataset: the **L2-ARCTIC** dataset (Zhao et al., 2018), which is a non-native English corpus containing 11026 utterances from 24 non-native speakers. Note that to evaluate mismatch localization, each annotated phoneme in the dataset needs to come with an onset and offset time label; to the best of our knowledge, the L2-ARCTIC dataset is currently *the only real-world dataset suitable for this problem setting.*

Mispronunciation Localization Results As shown in Table 2, similar to the results on Mismatch-AudioMNIST, Our ML-VAE models dramatically outperform Two-Pass-FA and w2v-CTC in all three metrics. Compared with original ML-VAE, the variant with reinforcement learning yields better localization performance of mispronunciation.

Table 2: Mispronunciation localization results on L2-ARCTIC.

Model	# Params	PR$_{\textbf{ML}}$%	RE$_{\textbf{ML}}$%	F1$_{\textbf{ML}}$%
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)	24M	6.46	11.97	8.39
ML-VAE-RL (ours)	25M	**8.20**	**13.57**	**10.22**

Interestingly, though w2v-CTC is based on a powerful wav2vec 2.0 acoustic model, 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 which contains unknown content mismatches between the speech data and the corresponding text; w2v-CTC is fine-tuned on the speech data with mispronunciations, and therefore its recognition results tend 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.

Ablation Study We present ablation study results to demonstrate the effectiveness of our proposed learning algorithm. Table 3 shows the results of ML-VAE-RL optimized with $\hat{\mathcal{B}}$ instead of $\bar{\mathcal{B}}$. The first row shows that directly using $\hat{\mathcal{B}}$ leads to very poor mispronunciation localization performance. The second row shows that our proposed optimization method effectively improves upon the traditional joint optimization method.

Table 3: Ablation study results.

Model	PR$_{\mathbf{ML}}$%	RE$_{\mathbf{ML}}$%	F1$_{\mathbf{ML}}$%
Ablation 1: ML-VAE-RL using $\hat{\mathcal{B}}$ instead of $\bar{\mathcal{B}}$ for optimization	2.31	1.53	1.84
Ablation 2: ML-VAE-RL w/ joint optimization	4.90	2.89	3.64
Ablation 3: ML-VAE-RL w/ $\hat{\mathcal{B}}$ & $\hat{\mathcal{Y}}$ estimated separately	7.31	10.85	8.73
ML-VAE (ours)	6.46	11.97	8.39
ML-VAE-RL (ours)	**8.20**	**13.57**	**10.22**

The third ablation study compares our estimation method with a traditional one, which estimates $\hat{\mathcal{B}}$ and $\hat{\Pi}$ separately based on a single-path FSA. Details on this traditional algorithm can be found in Appendix A. Compared with this ablation model, our proposed ML-VAE-RL yields better results. This proves the effectiveness of our proposed estimation algorithm, which is specifically designed for the mismatch localization task.

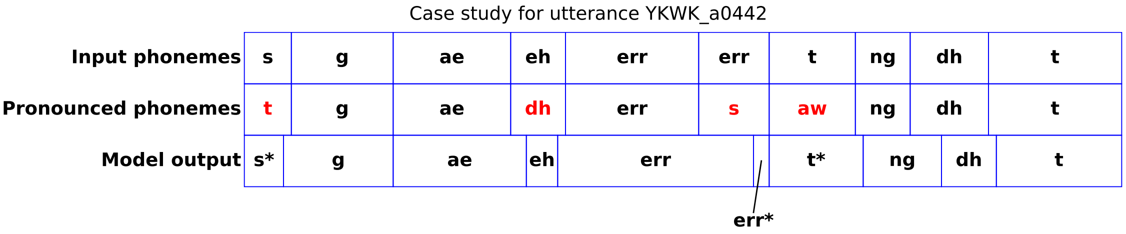

Figure 5: Case study for the last ten phonemes of the utterance YKWK_a0442. The first and second rows show the input phonemes and the actual phonemes pronounced by the speaker, respectively. The last row shows ML-VAE's predicted mispronunciation localization result.

A Case Study Besides quantitative analysis, we also provide a case study to showcase ML-VAE-RL in Fig. 5. Each rectangle in Fig. 5 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-RL's mispronunciation localization result. In the last row, a phoneme with a star sign (e.g. 't*') indicates that ML-VAE-RL 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. We also perform case study for the ablation models, which can be found in Appendix B.

10 Conclusions

In this work, we present a hierarchical Bayesian deep learning model to address the mismatch localization problem in cross-modal sequential data. More specifically, two variants are proposed, namely ML-VAE and ML-VAE-RL. 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 powerful wav2vec 2.0 acoustic model. We also perform ablation studies to verify the effectiveness of our training algorithm.

References

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.

Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Interpreting and Explaining Deep Neural Networks for Classification of Audio Signals. 7 2018. URL `http://arxiv.org/abs/1807.03418`.

Peter Bell and Steve Renals. A system for automatic alignment of broadcast media captions using weighted finite-state transducers. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 675–680. IEEE, 2015.

Germán Bordel, Mikel Penagarikano, Luis Javier Rodriguez-Fuentes, and Amparo Varona. A simple and efficient method to align very long speech signals to acoustically imperfect transcriptions. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

Norbert Braunschweiler, Mark J F Gales, and Sabine Buchholz. Lightly supervised recognition for automatic alignment of large coherent speech recordings. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

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*, pp. 2980–2988, 2015.

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.

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

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

Michael Finke and Alex Waibel. Flexible transcription alignment. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pp. 34–40. IEEE, 1997.

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

Timothy J Hazen. Automatic alignment and error correction of human generated transcripts for long speech recordings. In *Ninth International Conference on Spoken Language Processing*, 2006.

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*, pp. 1878–1889, 2017.

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

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.

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.

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

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

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.

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.

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

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

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

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.

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

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

Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech \& Language*, 16(1):69–88, 2002.

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

Pedro J Moreno and Christopher Alberti. A factor automaton approach for the forced alignment of long speech recordings. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4869–4872. IEEE, 2009.

Pedro J Moreno, Christopher F Joerg, Jean-Manuel Van Thong, and Oren Glickman. A recursive algorithm for the forced alignment of very long audio segments. In *ICSLP*, volume 98, pp. 2711–2714, 1998.

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

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*, pp. 2025–2031, 2016.

Adriana Stan, Peter Bell, and Simon King. A grapheme-based method for automatic alignment of speech and text data. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pp. 286–290. IEEE, 2012.

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

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

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

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

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

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*, pp. 2048–2057. PMLR, 2015.

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.

A 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.

A.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 correctness of the pronunciation.

When estimating $\hat{\mathcal{B}}$, we adopt the single-path FSA. Fig. 6 demonstrates the partial FSA 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). The sentence-level FSA can be constructed by connecting all phoneme-level FSA together, according to the input phoneme sequence $\mathcal{C}$.

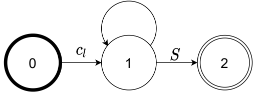

Figure 6: Single-path FSA.

Based on such an FSA, the following MAP estimator is used to estimate $\hat{\mathcal{B}}$:

$$\hat{\mathcal{B}} = \underset{\mathcal{B}}{\operatorname{argmax}} \prod_{t=1}^{T} p(y_t|y_1, ..., y_{t-1}) \frac{q_{\phi_p}(y_t|x_t)}{p(y_t)}. \tag{11}$$

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

$$p(y_t|y_1, ..., y_{t-1}) = \begin{cases} p(b_t = 0), & \text{if } y_t = y_{t-1} \\ p(b_t = 1), & \text{if } y_t \neq y_{t-1}, \end{cases} \tag{12}$$

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.

A.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:

$$\hat{\pi}'_l = \underset{\pi'_l}{\operatorname{argmax}} \prod_{t=u}^{v} p(y_t|y_1, ..., y_{t-1}) \frac{q_{\phi_p}(y_t|x_t)}{p(y_t)} = \underset{\pi'_l}{\operatorname{argmax}} \begin{cases} p(\pi_s = 0) \prod_{t=u}^{v} \frac{q_{\phi_p}(y_t = c_l|x_t)}{p(y_t = c_l)}, & \text{if } \pi'_l = 0 \\ p(\pi_s = 1) \prod_{t=u}^{v} \frac{q_{\phi_p}(y_t \neq c_l|x_t)}{p(y_t \neq c_l)}, & \text{if } \pi'_l = 1, \end{cases} \tag{13}$$

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 $p(y_t \neq c_l) = 1 - p(y_t = c_l)$.

B Case Study of Ablation Models

Fig. B shows the case study with the outputs of three ablation models presented in Table 3. It is shown that the joint optimization model yields the worst outputs because the poor alignment results. Compared with the other two ablation models, ML-VAE clearly outperforms them in terms of localizing the mispronounced phonemes.

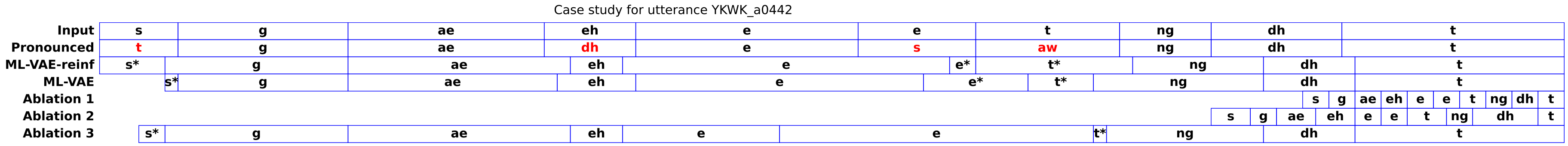
Case study for utterance YKWK_a0442
Input s g ae eh e e t ng dh t
Pronounced t g ae dh e s aw ng dh t
ML-VAE-reinf s* g ae eh e e* t* ng dh t
ML-VAE s* g ae eh e e* t* ng dh t
Ablation 1 s g ae eh e e t ng dh t
Ablation 2 s g ae eh e e t ng dh t
Ablation 3 s* g ae eh e e t* ng dh t

Figure 7: 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 third row shows ML-VAE's predicted mispronunciation localization result. The last three rows show the outputs of the three ablation models as presented in Table 3. To save space, the phoneme **err** is denoted as **e** in this figure.