Layer-Wise Fast Adaptation for End-to-End Multi-Accent Speech Recognition

Yanmin Qian, *Senior Member, IEEE*, Xun Gong, and Houjun Huang, *Member, IEEE*

***Abstract*—The variety and complexity of accents pose a huge challenge to robust Automatic Speech Recognition (ASR). Some previous work has attempted to address such problems, however most of the current approaches either require prior knowledge about the target accent, or cannot handle unseen accents and accent-unspecific standard speech. In this work, we aim to improve multi-accent speech recognition in the end-to-end (E2E) framework with a novel layer-wise adaptation architecture. Firstly, we propose a robust deep accent representation learning architecture to obtain accurate accent embedding, and some advanced schemes are designed to further boost the quality of accent embeddings, including phone posteriorgram (PPG) feature, TTS based data augmentation in the training stage, test-time augmentation and multi-embedding fusion in the testing stage. Then, the layer-wise adaptation with accent embeddings is developed for fast accent adaptation in ASR, and two types of adapter layers are designed, including the gated adapter layer and multi-basis adapter layer. Compared to the usual two-pass adaptation, these adapter layers are injected between the ASR encoder layers to encode the accent information in ASR flexibly, and perform fast adaption on the corresponding speech accent. The experiments on Accent AESRC corpus show that the proposed deep accent representation learning can capture accurate accent knowledge, and get high performance on accent classification. The new layer-wise adaptation architecture with the accurate accent embedding outperforms the other traditional methods, and obtains consistent ∼15% relative word error rate (WER) reduction on all kinds of testing scenarios, including seen accents, unseen accents and accent-unspecific standard speech.**

***Index Terms*—End-to-end speech recognition, multi-accent, layer-wise adaptation, accent embedding.**

I. Introduction

ONE of the challenges for ASR today is the support for multiple accents, which is often referred as multi-accent speech recognition in the literature. Distinguished by phonology, grammar, vocabulary, and orthography (e.g. "color" vs "colour"), accents are usually divided and named by social groups and geographical regions [1], [2]. For example, The people in China, the United Kingdom and the United States speak English with different accents, which are referred to as the Chinese accent, British accent, and American accent, respectively. Although state-of-the-art ASR systems report high performance, their performance still degrades dramatically when processing accented data. The main difficulties of multi-accent ASR are: (1) the lack of accented data in training which causes the unbalance between accent-unspecific and accented data [3], (2) the catastrophic forgetting where performance on the standard data decreases dramatically, during fine-tuning on accented data [4].

One straightforward way is to feed the accent utterance into accent-specific model [5], where the accent-specific model is fine-tuned on the corresponding accented data. Usually, the accent of an utterance is known as prior information, and if no accent label is provided in advance, an auxiliary accent identifier is applied [6], [7]. However, building accent-specific models is cumbersome in production and increases maintenance costs. Accordingly some techniques are proposed to adapt a unified model to perform well on all different accents. KL-divergence [8], [9], [10] between the fine-tuned and baseline models is applied through weight constraint adaptation or elastic weight consolidation. Compared with the fine-tuning scheme on specific accent, such methods try to avoid catastrophic forgetting, but lack of improvements on accented data.

Recently multi-task learning (MTL) is proposed to jointly optimize the objective of accent identification and speech recognition by sharing some shallow layers, and improvements on accented data can be observed. Yang et al. [11] shares the encoder with separate output layers for two accents and an accent identifier branches from the encoder to make a choice of which output layer to use. Jain et al. [12], [13] adds two auxiliary tasks, i.e. accent identification and phone identification, into the ASR task. Li et al. [14] and Rao et al. [15] insert special accent label tokens to the text label sequences during training inspired by code-switching ASR. Hu et al. [16] gives a theoretical guarantee on how accent adversarial training learns accent-invariant representations. However, such approaches have a large number of parameters to estimate with back-propagation and cannot always avoid overfitting in case of limited accented data. Usually the improvement is also not large.

Inspired from the success of speaker adaptation in ASR, some adaption approaches are developed to incorporate accent information into a single generic ASR model. Mixture of experts (MoE) is one method proposed in [13], [17], [18]. Different experts are specialized to operate on one aspect in the input

Manuscript received 18 September 2021; revised 21 March 2022 and 22 July 2022; accepted 24 July 2022. Date of publication 15 August 2022; date of current version 9 September 2022. This work was supported in part by China NSFC Projects under Grants 62122050 and 62071288 and in part by Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Lei Xie. *(Corresponding author: Yanmin Qian.)*

Yanmin Qian and Xun Gong are with the X-LANCE, Department of Computer Science and Engineering and MoE Key Laboratory of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yanminqian@gmail.com; gongxun@sjtu.edu.cn).

Houjun Huang is with the AISpeech Ltd, Suzhou 215000, China (e-mail: houjun.huang@aispeech.com).

Digital Object Identifier 10.1109/TASLP.2022.3198546

space. Outputs of the experts are then linearly combined using data-dependent weights generated by an additional auxiliary accent identifier. Selection of experts by one-hot encoding of accents is another intuitive technique outperforming the accent-specific models, and each expert is designed to cover certain types of accents [14], [19]. Yoo et al. [17] and Zhu et al. [20] build upon the one-hot approach by further integrating accent information using Feature-wise Linear Modulation (FiLM). To solve the 'unknown' accent issue, [17] tried to add an 'unknown' accent. However the drawbacks of one-hot accent encoding are that it is not effective for unseen accents or accent-unspecific data and it usually requires auxiliary accent labels.

In this paper, we proposed a layer-wise adaptation mechanism to improve the speech recognition accuracy on the accented speech data. This is an extension of our prior work [21], but with more contents, experiments and analysis. Firstly to obtain expressive accent representation, a neural network based deep accent representation learning framework is developed with both TDNN [22] and Ecapa-TDNN [23], and some advanced schemes are also applied to generate better accent embeddings, including posteriorgram (PPG) input features, text-to-speech (TTS) based training data augmentation, test-time augmentation on the test phase and multiple embedding fusion. Instead of usual concatenating accent embeddings with input features for the ASR system, a layer-wise adaptation using accent embedding is then proposed to fast adapt the speech recognition system on the accented data. The gated mechanism and multi-basis combination in one adapter layer are rapidly adjusted with the accent embeddings. With the proposed architecture, accent adaptation can be performed in a flexible manner, and achieve an improved system performance with a compact adaptation parameter set. Moreover, as the proposed architecture models different accents in the continuous embedding space, it can naturally cope with unseen accents and avoid the performance degradation on the accent-unspecific standard speech data.

The main contributions of this paper can be summarized as follows:

- A deep accent representation learning framework is developed on the advanced deep models, and some useful strategies are further applied upon this framework to generate more accurate and effective accent embedding representation.
- A rapid layer-wise adaptation architecture with accent embedding is proposed for accented speech recognition, and this architecture works well on all kinds of end-to-end ASR models.
- Two types of adapter layers are designed, one utilizes a gated mechanism and the other consists of multiple bases. The accent information can be learned flexibly with these adapter layers, and these two types can be also integrated together to get more improved performance.
- The entire proposed method is significantly better than traditional methods on accented speech recognition which obtains substantial ASR performance improvement on seen and unseen accents. Meanwhile, the proposed adaptation method succeed in avoiding catastrophic forgetting on accent-unspecific standard speech, compared to other methods.

The rest of the paper is organized as follows. Basic end-to-end ASR models are firstly reviewed in Section II. Section IV presents a novel deep accent representation learning framework with some useful strategies to generate better accent embeddings, and Section IV proposes the layer-wise fast adaptation architecture with accent embedding for multi-accent speech recognition. The detailed experimental setup, results and analysis are described in Sections V and VI, and finally the conclusions are given in Section VII.

II. End-to-End ASR Architecture

This section reviews the joint CTC/Attention architecture [24], [25], [26], which takes advantages of both CTC and attention-based end-to-end ASR approaches during training and decoding, and this architecture is also mainly utilized in this work.

A. Connectionist Temporal Classification (CTC)

CTC [27] enforces a monotonic mapping from feature $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T]^\intercal, \boldsymbol{x}_t \in \mathcal{R}^F$ to token sequence $\boldsymbol{C} = [c_1, c_2, \ldots, c_L]^\intercal, c_l \in \mathcal{U}$, where T is the frame length, F is the feature dimension, L is the sequence length, $\boldsymbol{x}_t$ is a D-dimensional acoustic vector at frame t, and c_l is the token at position l.

CTC introduces a many-to-one function from frame-wise latent variable sequences to token predictions with shorter lengths. With several conditional independence assumptions, the posterior distribution, $p(\boldsymbol{C}|\boldsymbol{X})$, is represented as follows:

$$p_{ctc}(\boldsymbol{C}|\boldsymbol{X}) = \sum_{\boldsymbol{Z}} \prod_{t} p(z_t|\boldsymbol{X}), \tag{1}$$

where $\boldsymbol{Z} = [z_1, z_2, \ldots, z_T]^\intercal,\ z_t \in \mathcal{U} \cup \{\text{blank}\}$ and $p(z_t|\boldsymbol{X})$ is a frame-wise posterior distribution. By deleting blank symbol and merging same labels in $\boldsymbol{Z}$, $\boldsymbol{C}$ can be achieved. CTC preserves the benefits of enforcing the monotonic behavior of speech-label alignments, and also avoids the HMM/GMM construction step or preparation of pronunciation dictionary.

B. Attention-Based Encoder-Decoder

As one of the most commonly used modeling techniques in ASR, the attention-based framework [24] encodes speech features into a fixed dimension vector representation $\boldsymbol{H}$, which is then consumed by the decoder to produce a distribution over the outputs. We can directly estimate the posterior distribution $p(\boldsymbol{C}|\boldsymbol{X})$ using the chain rule:

$$p_{att}(\boldsymbol{C}|\boldsymbol{X}) = p(\mathrm{c}_1|\boldsymbol{X}) \prod_{\mathrm{l}=2}^{\mathrm{L}} \mathrm{p}(\mathrm{c}_\mathrm{l}|\mathrm{c}_1, \ldots, \mathrm{c}_{\mathrm{l}-1}, \boldsymbol{X}). \tag{2}$$

Typically, a RNN or self-attention based encoder transforms the speech feature $\boldsymbol{X}$ into frame-wise hidden vector $\boldsymbol{H}$:

$$\boldsymbol{H} = \mathrm{Encoder}(\boldsymbol{X}), \tag{3}$$

where $\boldsymbol{H} \in \mathcal{R}^{A \times T'}$, A is the feature dimension of encoder and $T' < T$ is the length of $\boldsymbol{H}$ due to the sub-sampling technique. The decoder estimates the posterior probability distribution over the previous letters with the attention mechanism, given $\boldsymbol{C}_{<l}$:

$$p_{att}(c_l|\boldsymbol{X}) = \text{Decoder}(\text{Attention}(\boldsymbol{H}, \boldsymbol{d}_{<l}), \boldsymbol{d}_{<l}, \boldsymbol{C}_{<l}), \quad (4)$$

where $\boldsymbol{d}_{<l}$ is the previous output from the decoder. For the recurrent neural network (RNN) based decoder, the previous letters $\boldsymbol{C}_{<l}$ are encoded as latent context vectors. For the transformer-based decoder, the previous letters are inputted into the decoder together.

C. Joint CTC/Attention

The joint CTC/Attention architecture [25] benefits from both CTC and attention-based models since the attention-based encoder-decoder is trained together with CTC within the multi-task learning (MTL) framework. The encoder is shared across CTC and attention, and the loss function is a logarithmic linear combination of the CTC and attention posterior probability:

$$\mathcal{L}_{jca} = -\lambda_{ctc} \log p_{ctc}(\boldsymbol{C}|\boldsymbol{X}) - (1 - \lambda_{ctc}) \log p_{att}(\boldsymbol{C}|\boldsymbol{X}), \quad (5)$$

where λ_{ctc} is a tunable scalar satisfying $\lambda \in [0, 1]$ and $p_{att}(\boldsymbol{C}|\boldsymbol{X})$ here is an approximated probability conditioned on previous labels.

For decoding, the joint decoded token c_l at position l can be predicted as:

$$\hat{c}_l = \arg\max_{c_l \in \mathcal{U}} \lambda_{ctc} s_{ctc}(c_l) + (1 - \lambda_{ctc}) s_{att}(c_l), \quad (6)$$

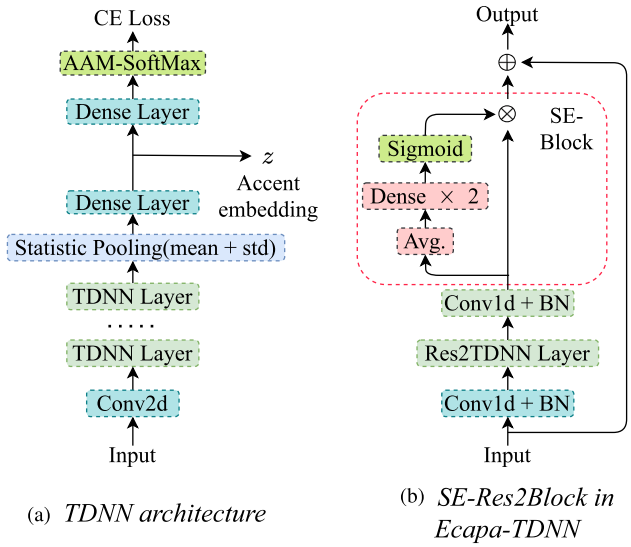
where $s_{att}(c_l) \triangleq \log p_{att}(c_l|\hat{\boldsymbol{C}}_{<l}, \boldsymbol{X})$ is the attention score and $s_{ctc}(c_l) \triangleq \log p_{ctc}(\hat{\boldsymbol{C}}_{<l}, c_l|\boldsymbol{X})$ is the CTC prefix score [26]. The attention and CTC scores $s_{att}(l)$ can be accumulated recursively from hypothesis scores from one step before and beam search method is utilized.

III. Deep Accent Representation Learning

Deep representation learning using deep models has been proven useful on some tasks, such as speaker verification with d-vector [28], x-vector (TDNN) [22] and r-vector (ResNet) [29], [30]. Motivated by this, accent representation learning with advanced deep models is firstly designed in this work to obtain accurate accent embeddings. Then some useful technologies, including the phone posteriorgram (PPG) feature, TTS-based training data augmentation, test-time augmentation and multi-embedding fusion are proposed to boost the quality of the accent representations.

A. Deep Accent Representation Learning

Motivated by the speaker embedding learning using deep models [22], [28], we also utilize the deep model to do the accent representation learning. Time delay neural network (TDNN) based architecture [22] is introduced to get effective accent embeddings at first, illustrated in Figure 1(a). The usual acoustic features, such as Fbank or MFCC, can be firstly fed through several frame-level layers in TDNN, and then a statistics pooling layer is adopted to aggregate the frame-level deep features into a segment-level statistical matrix. The outputs of one dense layer are extracted as the accent embeddings $\boldsymbol{z}$, which are considered to encode the useful accent-related knowledge. To enforces the similarity of intra-class samples and the diversity of inter-class samples, several softmax-based loss functions have been proposed [31], [32], a thorough comparison of different loss functions was carried out in [31] on speaker verification task. Here we choose the best-performing of them, the additive angular margin softmax (AAM-softmax) [33] loss as the objective function, as accent identification is analogous to speaker verification. For a raw wav with accent label $a \in \{1, 2, \ldots, N\}$, the d-dimensional accent embedding $\boldsymbol{z} \in \mathcal{R}^d$ is firstly accessed by deep model like TDNN, where N is the number of accents. The prediction is then generated as:



(a) *TDNN architecture* (b) *SE-Res2Block in Ecapa-TDNN*

Fig. 1. The proposed deep accent representation learning framework. (a) Accent embedding with time-delay neural network (TDNN), and (b) Accent embedding with Ecapa-TDNN using SE-Res2Block structure. $\oplus$ is the residual connection, and $\otimes$ denotes the channel-wise multiplication.

$$\hat{\boldsymbol{Y}} = \boldsymbol{W}^T * \boldsymbol{z} + \boldsymbol{b}, \quad (7)$$

$$\mathcal{L}_{softmax} = -\log \frac{e^{\hat{Y}_i}}{\sum_{j=1}^{N} e^{\hat{Y}_j}}, \quad (8)$$

where $\hat{\boldsymbol{Y}} \in [0, 1]^N$, $\boldsymbol{Y}_i = \begin{cases} 1, i = a \\ 0, else \end{cases}$ are prediction, ground truth. $\boldsymbol{W} \in \mathcal{R}^{d \times N}$, $\boldsymbol{b} \in \mathcal{R}^N$ are the weight and bias in the projection layer. Denote $\theta_a = \arccos(\boldsymbol{W}_a, \boldsymbol{z})$ be the angle between $\boldsymbol{W}_a$ and $\boldsymbol{z}$, $\boldsymbol{W}_a$ is the a-th column of $\boldsymbol{W}$, the modified AAM-softmax loss is:

$$\mathcal{L}_{\text{AAM-softmax}} = -\log \frac{e^{s\cos(\theta_a + m)}}{e^{s\cos(\theta_a + m)} + \sum_{j=1, j \neq a}^{N} e^{s\cos(\theta_j)}}, \quad (9)$$

where m, s are hyper-parameters, m is the additive margin and s is the scale parameter which can help the model converge faster. To stabilize the convergence of the training, the margin is gradually increased from 0 to the m_{final}.

In addition to TDNN, more advanced Ecapa-TDNN[23] is also used. Ecapa-TDNN can improve the time-delay layer into 1-dimensional squeeze-excitation (SE) [34] res2block shown in Figure 1(b). The SE res2block is integrated by residual connection to model global channel inter-dependencies. While statistics pooling, channel- and context-dependent self-attention mechanism are adopted. The squeeze operation is a simple mean vector of the feature $\boldsymbol{F} \in \mathcal{R}^{C\times T}$, denoted as mean$(\boldsymbol{F})$ across time domain. And the subsequent excitation operation calculates a weight $\boldsymbol{w}$ for each channel as follows:

$$\boldsymbol{w} = \sigma(\boldsymbol{W}_2 * f(\boldsymbol{W}_1 * \text{mean}(\boldsymbol{F}) + \boldsymbol{b}_1) + \boldsymbol{b}_2), \tag{10}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $f(\cdot)$ denotes ReLU function, and $\boldsymbol{W}_1 \in \mathcal{R}^{R\times C}, \boldsymbol{W}_2 \in \mathcal{R}^{C\times R}$, C is the channel number and R refers to the reduced dimensionality. Finally $\boldsymbol{w}$ is applied to the original input $\boldsymbol{F}$ through channel-wise multiplication $\otimes$: $\boldsymbol{F}_{\text{output}} = \boldsymbol{w} \otimes \boldsymbol{F}$.

B. Hybrid ASR based PPG Features

Although Fbank and MFCC spectrum features are widely used in speech processing, they may not be the most appropriate for accent representation learning. One reason is that the models trained with Fbank or MFCC could not take advantage of the data with accent labels. Another is that spectrum features are not task-oriented and may contain some nuisance attributes like speaker or text information, which makes the accent representation learning harder.

Accordingly, phone posteriorgram (PPG) features are explored here to do better accent embedding learning. PPG features are first used for voice conversion [35], [36], and it is a time-versus-class vector that represents the posterior probabilities of phonetic classes at the frame level. In this work to make PPG features speaker-independent, we firstly train a hybrid speaker-independent ASR model with both the accented training data and the accent-unspecific standard data, and then extract the PPG features from this SI-ASR acoustic model. In addition to the original static PPG features, the dynamic PPG features can also be utilized with the first- and second-order differences. With this process, the resulting PPG features have the speaker-independent property which helps improve the robustness of the accent embedding learning system.

C. TTS based Data Augmentation

With limited accented training data, building a more robust accent representation model requires data augmentation schemes. Besides the conventional methods, such as speed or volume perturbation, adding noises and room impulse responses, we develop a novel accent-specific TTS-based data augmentation to generate more accented data for accent representation learning. The high-quality synthesized speech data can show speaker variations within same texts and indistinguishable from human speech, and is proven useful in speaker verification [37] and ASR [38]. In this work, we choose FastSpeech [39] as our synthesizer and LPCNet [40] as the vocoder. FastSpeech is a transformer-based TTS model which generates the entire mel-spectrogram from text in parallel. LPCNet is a variant of WaveRNN that combines linear prediction with recurrent neural networks to significantly improve the efficiency of speech synthesis.

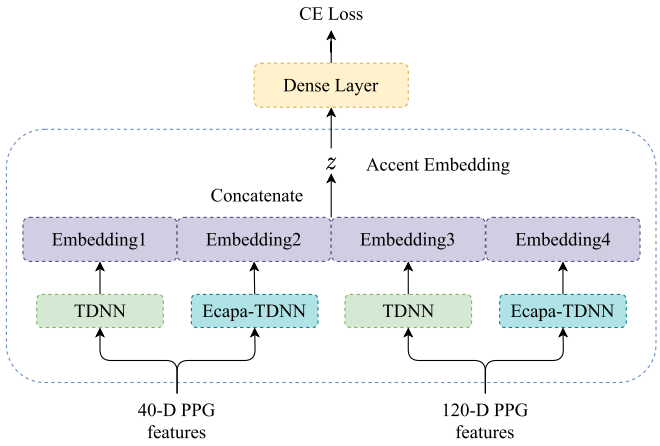


Fig. 2. The multi-embedding fusion model includes the TDNN and Ecapa-TDNN sub-models with two PPG feature types. '40D-PPG' features denote the 40-dimensional PPG features, while '120D-PPG' features represent the original static 40D-PPG features with their first and second differentials.

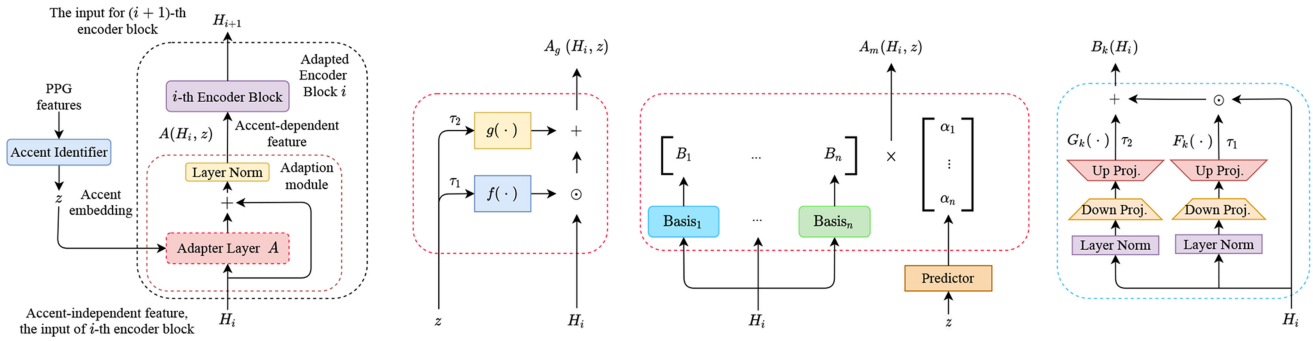
To train a universal FastSpeech synthesizer, we first train a TDNN speaker model and then feed speaker embedding and phoneme representations to train the synthesizer. After that, we train accent-specific synthesizers by fine-tuning to capture the characteristics of each accent. As for LPCNet vocoders, male and female speakers are trained separately. While synthesizing, we use speaker-specific x-vectors with randomly selected texts to generate new accented by the accent-specific synthesizer, thus can not only preserve the speaker style but also adapt to a different accent.

D. Test-Time Augmentation

In addition to the data augmentation on the training stage to enlarge the training data, e.g. the above TTS-based data augmentation, the augmentation on the test sample is also designed, which is named Test-Time Augmentation (TTA) in this work. Test-time augmentation is a common trick proved useful in image classification tasks to improve the test accuracy [41]. Instead of only predicting the label of the original test audio, the model takes multiple augmented versions of the test audio to give the final result. Here we augment the test speech by speed perturbation and then splice the augmented audios into one longer utterance, which is then used to extract the accent embeddings.

E. Multi-embedding Fusion

Finally we use a multi-embedding fusion scheme to achieve final accent representation and identification, which is shown as Figure 2. Four separate accent classifiers are trained individually, and they use different deep models and different PPG features respectively. Each model generates the related accent embedding, and then four types of accent embeddings are combined to form a new multi-embedding fusion representation. This



(a) *Layer-wise Adaptation in the i-th Encoder Block* (b) *Gated Adapter Layer $\mathcal{A}_g$* (c) *Multi-Basis Adapter Layer $\mathcal{A}_m$* (d) *Basis B_k in $\mathcal{A}_m$*

Fig. 3. Schematic diagram of the proposed adapter layer in the proposed layer-wise fast adaptation architecture. The proposed adapter layer in (a) is optionally inserted in each encoder block of the End-to-end ASR model. (b) is the proposed gated adapter layer $\mathcal{A}_g$, and (c) is the proposed multi-basis adapter layer $\mathcal{A}_m$ with the Basis B_k in (d). Here, $+$, $\times$, and $\odot$ denote summation, matrix multiplication, and element-wise product, respectively.

combined multi-embedding representation can be more robust and accurate, and it can be utilized in the accent adaptation for speech recognition or usual accent identification.

IV. Layer-Wise Fast Adaptation on End-to-End Multi-Accent ASR

The speech recognition model usually lacks generalization on accented data due to the accent mismatch issues. In this section, a novel layer-wise adaptation architecture with above accent embedding is proposed to improve multi-accent speech recognition while avoiding catastrophic forgetting on the non-accented data.

A. The Adapter Layer in Layer-Wise Adaptation

The architecture of the new ASR encoder with the proposed adapter layer is illustrated in Fig. 3(a). Denoted as $\mathcal{A}$, the adapter layer is actually a transformation which is injected into the encoder blocks to transform the accent-independent representations $\boldsymbol{H}$ into the accent-dependent representations $\mathcal{A}(\boldsymbol{H}, \boldsymbol{z})$. At i-th encoder block, the transformation takes the representation $\boldsymbol{H}_i \in \mathcal{R}^{A \times T'}$ and the accent embedding $\boldsymbol{z} \in \mathcal{R}^d$ as the input, where d is the dimension of accent embedding $\boldsymbol{z}$, and then wraps the output $\mathcal{A}(\boldsymbol{H}_i, \boldsymbol{z})$ into later part in i-th encoder block by a residual connection, shown as "+" in Fig. 3(a), to ensure the original acoustic information to flow through later encoder layers. Additionally, we normalize the input of each adapter basis by LayerNorm [42]. Note that the accent embedding $\boldsymbol{z}$ is obtained using the strategy proposed in the above section. Two different types of adapter layers are explored in this work, including the gated adapter layer $\mathcal{A}_g$ and multi-basis adapter layer $\mathcal{A}_m$, which will be described in the following subsections. Moreover, we also explore different encoder block modules like Transformer [43] and convolution augmented transformer (Conformer) [44] to verify the generalization of the proposed architecture in the experiments.

B. Gated Adapter Layer

The first type of adapter layer is designed to attain the transform function by a gate mechanism, denoted as $\mathcal{A}_g$. The gated adapter layer encodes the accent embedding $\boldsymbol{z}$ into the hidden representations. As illustrated in Fig. 3(b), a scaling factor function $f(\boldsymbol{z})$ and a shifting factor function $g(\boldsymbol{z})$ can be optionally applied to adapt the input acoustic features to accent-dependent space:

$$\mathcal{A}_g(\boldsymbol{H}_i, \boldsymbol{z}) = \tau_1 f(\boldsymbol{z}) \odot \boldsymbol{H}_i + \tau_2 g(\boldsymbol{z}), \tag{11}$$

where $\mathcal{A}_g$ is the gated adapter layer, $\odot$ denotes the element-wise product. τ_1, τ_2 are boolean matrices to decide the trigger of $f(\cdot)$ and $g(\cdot)$, so that scaling-only and shifting-only are alternative transformations. For example, $\tau_1 = \mathbf{1}, \tau_2 = \mathbf{0}$ for scaling-only case. $f(\boldsymbol{z})$ and $g(\boldsymbol{z})$ are separately generated by a single dense layer with $\sigma = \tanh(\cdot)$ activation:

$$\begin{aligned} f(\boldsymbol{z}) &= \sigma(\boldsymbol{W}_f \boldsymbol{z} + \boldsymbol{b}_f), \\ g(\boldsymbol{z}) &= \sigma(\boldsymbol{W}_g \boldsymbol{z} + \boldsymbol{b}_g), \end{aligned} \tag{12}$$

where $\boldsymbol{W}_f, \boldsymbol{W}_g \in \mathcal{R}^{d \times d}$ and $\boldsymbol{b}_f, \boldsymbol{b}_g \in \mathcal{R}^d$. Compared with explicit concatenation with accent embeddings, the gated adapter layer provides a 'soft' transformation by providing accent information at the input level of the encoder.

C. Multi-Basis Adapter Layer

Denoted as $\mathcal{A}_m$, the other adapter layer type is the multi-basis adapter layer in Fig. 3(c), which extracts accent information from the hidden representations $\boldsymbol{H}_i$ and then re-combines it with hidden representations. $\mathcal{A}_m$ is comprised of multiple bases B_k, $k \in \{1, 2, \ldots, n\}$. Each basis takes $\boldsymbol{H}_i$ as input, and the output is denoted as $B_k(\boldsymbol{H}_i)$. The output $B_k(\boldsymbol{H}_i)$ from each basis is linearly interpolated by the corresponding weight vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^\intercal$:

$$\mathcal{A}_m(\boldsymbol{H}_i, \boldsymbol{z}) = \boldsymbol{\alpha}^\intercal * \boldsymbol{B} = \sum_{k=1}^{n} \alpha_k B_k(\boldsymbol{H}_i), \tag{13}$$

where $k = 1, 2, \ldots, n$, and $n = \#\text{bases}$ is the number of adapter bases. Similar to Section IV-B, the scaling $F_k(\cdot)$ and shifting $G_k(\cdot)$ modules are used to transform the input $\boldsymbol{H}_i$ into the accent-dependent space as shown in Fig. 5(d):

$$B_k(\boldsymbol{H}_i) = \tau_1 F_k(\boldsymbol{H}_i) \odot \boldsymbol{H}_i + \tau_2 G_k(\boldsymbol{H}_i), \tag{14}$$

τ_1, τ_2 are boolean matrices to decide the trigger of $F_k(\cdot)$ and $G_k(\cdot)$.

We propose a sandglass-style structure for $F(\cdot)$ and $G(\cdot)$ modeling: a down-projection module and a up-projection module with non-linear activation ReLU$(\cdot)$. To make the bases in Fig. 3(d) simple and flexible, we can easily adjust the inner dimension between two projections depending on the complexity of the accented data.

The dynamical soft assignment of bases is done by the interpolation vector $\boldsymbol{\alpha} \in \mathcal{R}^n$. In this work, the accent embedding $\boldsymbol{z}$ is used to construct a predictor $\mathcal{P}(\cdot)$ module, which can estimate $\boldsymbol{\alpha}$ and give the guidance on the bases usage:

$$\boldsymbol{\alpha} = \text{Softmax}(\mathcal{P}(\boldsymbol{z})), \tag{15}$$

where $1 = \sum_{k=1}^{n} \alpha_k$ and the interpolation vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^\intercal$ are probabilities for multiple bases. The predictor $\mathcal{P}(\cdot)$ can be composed of several dense layers. The linear composition of the bases is designed to represent the final accent of each utterance.

D. Predictor Regularization on the Multi-Basis Adapter Layer

During the multi-basis adapter layer model optimization, we have found that, without any constraints, the shape of interpolation vector $\boldsymbol{\alpha}$ would rapidly reduce to one certain basis ($\#bases = 1$), which significantly limits the model ability on accent adaptation. This is caused by the mismatch between the distribution of predicted interpolation vectors and that of the expected ones at first. To circumvent this situation, an auxiliary task is proposed to regularize the predictor $\mathcal{P}$, so that it can be initialized randomly. The final objective loss combines the conventional ASR objective $\mathcal{L}_{jca}$ from Equation 5 and the element-wise mean square error (MSE) loss from predictor $\mathcal{P}$:

$$\mathcal{L}_{mtl} = \mathcal{L}_{jca} + \gamma_{mtl}\mathcal{L}_{\text{MSE}}(\boldsymbol{\alpha}_{gt}, \hat{\boldsymbol{\alpha}}), \tag{16}$$

where γ_{mtl} is a hyper-tuning parameter and $\boldsymbol{\alpha}_{gt}$ is the training label. As the purpose of this auxiliary task is to regularize each element of $\hat{\boldsymbol{\alpha}}$ to avoiding one-/two-hot vector, we can use pre-trained clustering labels with k-means method by minimizing the L2 square error:

$$Err = \sum_{i=1}^{n} \sum_{\boldsymbol{z} \in C_i} ||\boldsymbol{z} - \boldsymbol{\mu}_i||_2^2, \tag{17}$$

where $\boldsymbol{\mu}_i = \text{mean}_{\boldsymbol{z} \in C_i}(\boldsymbol{z})$ is the center of accent embeddings belonging to cluster C_i. Then the ground truth $\boldsymbol{\alpha}_{gt}$ can be estimated as:

$$\boldsymbol{\alpha}_{gt} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^\intercal, \text{ such that}$$
$$\alpha_k = \begin{cases} 1, & \text{if } k = \arg\min_i ||\boldsymbol{z} - \boldsymbol{\mu}_i||^2, \\ 0, & \text{else .} \end{cases} \tag{18}$$

V. Experimental Setup

A. Dataset

Our experiments are conducted on the Accented English Speech Recognition Challenge 2020 (AESRC2020) dataset [3][1] with the Librispeech corpus [45]. The sampling rate of these two datasets is 16 kHz. The AESRC2020 training set contains eight English accents, including England (UK), America (US), China (CHN), Japan (JPN), Russia (RU), India (IND), Portugal (PT), and Korea (KR). Each accent in the AESRC2020 dataset has 20 hours speech data collected from around 60 speakers. The auxiliary Librispeech set contains 1000 hours accent-unspecific speech of the audiobooks reading.

For evaluation, Librispeech test sets are used as the accent-unspecific standard English test data, consisting of dev-clean/other (dev c/o) and test-clean/other (test c/o) subsets. AESRC2020 test sets are used as the accented test data, consisting of a cross validation (cv) set and a test set. Note that the cv set in the AESRC2020 dataset has the same number of accents as the training set, while the test set has two more unseen accents compared to the training set, i.e. Canada (CAN) and Spain (ES). We present the word error rate (WER) on all above evaluation sets, including seen accents, unseen accents, and accent-unspecific standard speech.

B. ASR Configurations

The input feature of the ASR system is 80-dimensional log-Mel filterbanks (Fbank) with a 10 ms step size and a 25 ms window size. All features are extracted using the Kaldi toolkit [46] and applied with cepstral mean and variance normalization (CMVN) on the utterances. Target sequences are operated on 500 Byte-pair encoding [47] subword units. All models in our experiments have the same depth and a similar size, and this can be fair to do the comparison. The end-to-end (E2E) ASR model is built with ESPnet toolkit [48], and it comprises a transformer encoder and a transformer decoder joint with a CTC decoder. A two-layer CNN structure is used to down-sample acoustic features, and each of them has 512 filters with 3×3 kernel size and 2×2 stride, which shrinks the input sequence to $1/4$. The encoder has 12 transformer blocks, and the decoder has 6 transformer blocks, respectively. The multi-head attention module has 8 heads and the dimension of the query, key, and value is 512. The position-wise feed-forward module has the inner dimension of 2048. We choose Adam optimizer with a warmup learning rate schedule (25000 warm steps) to train our models (i.e. noam optimizer) [43]. SpecAugment [49] is applied for data augmentation during training. CTC joint parameter λ_{ctc} mentioned in Section II-C is set to 0.3. During inference, a rescore scheme [25] is applied with CTC weight of 0.3 and the beam width is 10 for beam search.

C. Deep Accent Representation Learning Configurations

For the accent representation learning, the Librispeech corpus is excluded, and only the 8 accents in the training set are

[1][Online]. Available: https://www.datatang.com/INTERSPEECH2020

TABLE I
ACCENT IDENTIFICATION ACCURACY (%) COMPARISON OF THE PROPOSED ACCENT EMBEDDING REPRESENTATION LEARNING WITH TDNN AND ECAPA-TDNN MODELS ON THE AESRC SET. 'I-VECTOR' DENOTES A LINEAR CLASSIFIER TAKES I-VECTOR AS INPUT. '+CONV' DENOTES THE CONVENTIONAL DATA AUGMENTATION, '+TTS' DENOTES THE PROPOSED TTS-BASED DATA AUGMENTATION, '+TTA' DENOTES THE PROPOSED TEST-TIME DATA AUGMENTATION, AND '+DELTA' DENOTES THE PPG FEATURES WITH THEIR FIRST AND SECOND DELTAS. 'FINAL SYSTEM' DENOTES THE SYSTEM USING ALL ABOVE TECHNIQUES AND FURTHER WITH MULTI-EMBEDDING FUSION

Model	Configuration	Accent CV Set								Accent	
		US	UK	CHN	IND	JPN	KR	PT	RU	CV Avg.	Test Avg.
I-vector		47.18	85.80	54.38	92.21	65.44	41.89	70.14	64.51	65.34	51.38
Challenge Baseline [3]		60.20	93.90	67.00	97.00	73.20	55.60	85.50	75.70	76.10	64.90
TDNN	Fbank	36.54	90.13	54.32	92.61	51.34	35.94	73.08	49.44	60.24	-
	PPG	78.89	92.28	81.20	98.71	76.01	83.61	84.47	76.79	83.74	-
	+CONV	77.07	91.84	86.78	99.39	77.96	87.11	86.88	78.65	85.57	-
	+TTS	80.43	93.04	94.48	99.54	81.72	90.33	90.90	87.19	89.74	-
	+TTA	80.79	92.98	94.87	99.70	83.94	88.55	91.96	89.29	90.32	-
	+DELTA	81.63	92.86	94.20	99.70	84.07	87.11	93.19	87.68	90.10	-
Ecapa-TDNN	Fbank	50.00	82.86	45.23	85.83	46.51	32.58	70.24	48.39	57.32	-
	PPG	82.12	91.71	81.99	98.93	77.96	82.44	86.94	76.11	84.51	-
	+CONV	81.56	92.17	89.68	99.31	79.44	88.54	88.74	83.04	87.71	-
	+TTS	79.19	93.23	94.25	99.31	81.45	90.47	93.25	87.25	89.86	-
	+TTA	80.86	93.17	94.70	99.77	84.95	86.76	93.32	87.19	90.15	-
	+DELTA	81.49	93.30	94.59	99.62	86.16	88.41	92.14	88.49	90.56	-
Final System		**82.68**	**93.36**	**95.37**	**99.77**	**84.88**	88.96	**93.69**	**89.85**	**91.13**	**83.63**

used. 40-dimensional and 120-dimensional PPG features are used in the accent identifier model training, and 40-dimensional Fbank feature is also applied as a comparison. The 120-dim PPG feature is the concatenation of 40-dim PPG features with their first and second derivatives. For TDNN [22] and Ecapa-TDNN [23] systems, the standard setup is applied. All systems are trained using AAM-softmax [33], and the weight-decay is applied of $2e^{-5}$. In addition, performance can be further improved when applying data augmentation on it. By employing speed perturb of $0.8\times, 0.9\times, 1.1\times, 1.2\times$, and adding noise, music, speech with MUSAN dataset [50] and room reverberation (room impulse responses, RIRs) from kaldi voxceleb recipe [46], we extend the training accented data to 10 times size. The model trained with these conventional augmented data approaches is denoted as 'CONV' in the following sections.

We then synthesize data with the TTS system as described in Section IV-C. The TDNN x-vector speaker model follows the settings in [22] trained with both accent and Librispeech data. The implementation of FastSpeech is based on the ESPnet toolkit [39], [48] and fine-tuned on each 20 hours accent data to build 8 synthesizers. The size of the input vocabulary is 41, including English phonemes, a pause break token, and a sentence boundary token. Additionally, we augment the decoder with a five-layer post-net [51] and the LPCNet vocoders are for the male and female speakers as a re-implementation based on [40]. To get more robust speaker-specific x-vectors, 30 utterances for each speaker are grouped to calculate the statistical x-vector. We use the speaker statistical x-vector and randomly selected reference texts to synthesize data with another accent synthesizer to generate additional three times as much data. For the test-time augmentation ('TTA'), each test utterance is augmented with $0.8\times, 0.9\times, 1.1\times, 1.2\times$ on time, and then aggregated to obtain the accent embedding.

VI. EXPERIMENTAL RESULTS

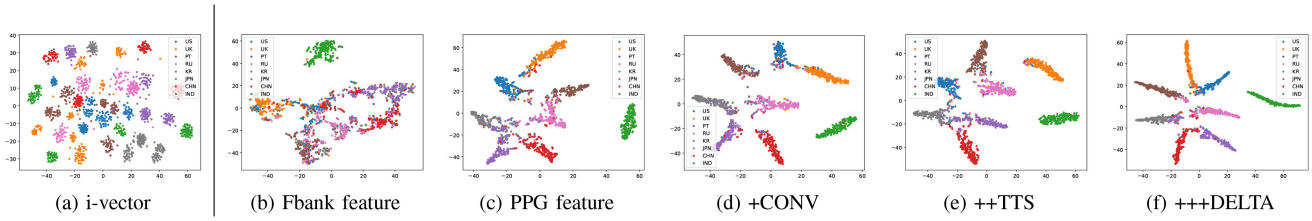
The detailed experimental results, comparison and analysis are described in this section. The results on accent representation learning are firstly presented and then those on the layer-wise adaptation with accent embedding for multi-accent speech recognition are given.

A. Evaluation on the Deep Accent Representation Learning

To evaluate the proposed methods for accent representation learning, we firstly construct the relative accent identifier and compare the accent identification accuracy. 'I-vector' system takes i-vector as input, and pass a 256-dim linear layer before the final layer to predict the accent label. 'Challenge Baseline' system directly takes the results from the dataset paper [3], which use Transformer as its base model. The higher accuracy indicates the better accent embedding representation. The system results and comparison are shown in Table I.

PPG vs. FBANK: The normal acoustic spectrum feature FABNK and the proposed accent PPG feature are performed on both TDNN and Ecapa-TDNN. It shows that the accent identification is not easy to do, and the deep model with usual FBANK can only achieve $\sim 60.0\%$ on accuracy. However, using the proposed PPG feature for accent representation, the accent identification can be boosted with a very large improvement, and the accent identification accuracy approaches 85.0%, which is $\sim 25.0\%$ absolutely better than FBANK.

Data Augmentation: Upon the system with PPG feature, the different data augmentation approaches are applied, including the conventional method modifying the raw wavform, the proposed TTS-based accent data augmentation and the test-time augmentation. The results show that all the data augmentation strategies are effective and improve the system performance significantly. The proposed TTS-based method is not conflict with



(a) i-vector (b) Fbank feature (c) PPG feature (d) +CONV (e) ++TTS (f) +++DELTA

Fig. 4. The 2-dimensional T-SNE illustrations of i-vector (the first one) and deep accent embeddings (others) on accent cv set, and 200 accent embeddings are randomly chosen for each accent. From left to right, different accent embedding learning strategies are compared.

conventional one, and still gets substantial gains. The test-time augmentation can obtain slight but consistent improvements on the accent embedding representation and identification.

Dynamic Delta Features: The dynamic feature is also important for speech processing, and the first and second delta features on PPG is still useful and get an improvement on most accents.

Multi-embedding Fusion: Finally, the proposed multi-embedding fusion scheme is evaluated, and it is performed with all above techniques. This is also the final accent embedding used in multi-accent ASR adaptation, and the accent identification results are shown as the last line of Table I. It shows that the final multi-embedding fusion system can achieve a promising performance position on accent identification. This system with the proposed accent embedding representation learning also ranks the first position on the accent identification track of AESRC challenge [3], [52], which is much better than the official challenge baseline shown as the first line of Table I.

Accent Embedding Visualization: To better understand the proposed accent representation learning, the t-SNE illustrations of different accent embeddings are shown and compared in Fig. 4. It obviously shows that i-vectors are scattered everywhere for the accents, and in contrast the proposed accent embeddings (except those using Fbank feature) are distinguished from each other clearly. This observation further demonstrates the effectiveness of the proposed deep accent representation learning. For the comparison of the accent embeddings using deep models, it is observed that the learned accent embedding from the Fbank system cannot be discriminated well, and most of the accents are overlapped with each other except Indian accent. In contrast, using the proposed PPG features, the clustering of each accent is much better, and the separation on different accents is much easier. Moreover the data augmentation and dynamic feature can further improve the accent embedding representation, and they can cluster better gradually. The last embedding representation achieves smaller intra-class distance and larger inter-class distance, and all accents are segregated well.

B. Evaluation on the Layer-Wise Accent Adaptation

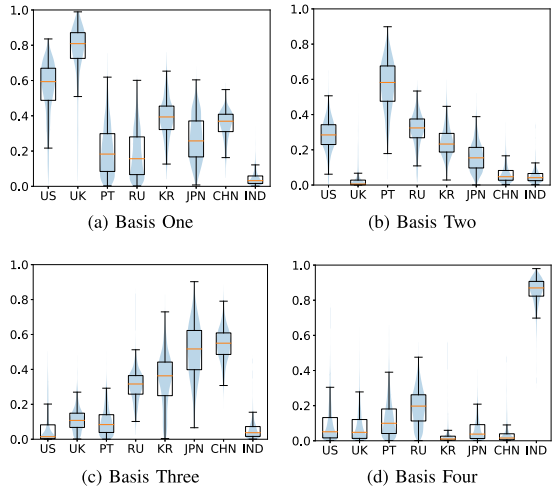
With utilization of the above accent embeddings, the proposed layer-wise adaptation architecture is then evaluated in this section. For the following experiments, the pre-trained transformer model is used as an initialization, and the adapter layers are then injected. The detailed configurations for the layer-wise adaptation are explored, including the type of the adapter layer, the position and number of the adapter layer, and the basis number in the multi-basis adapter layer. The results are shown in Table II.

TABLE II
PERFORMANCE (WER) (%) COMPARISON OF THE PROPOSED LAYER-WISE ACCENT ADAPTATION WITH DETAILED CONFIGURATION COMPARISON

Adapter Type	POS	$\#B_k$	Accent		Libri	
			cv	test	dev c/o	test c/o
Baseline			6.54	7.61	5.64/11.43	6.31/11.68
$\mathcal{A}_g$	{1}	-	5.89	6.89	5.27/10.39	5.76/10.79
$\mathcal{A}_g$	{2}	-	**5.87**	**6.77**	5.05/10.22	**5.53**/10.72
$\mathcal{A}_g$	{4}	-	5.90	6.78	**5.02/10.15**	5.59/**10.70**
$\mathcal{A}_g$	{10}	-	6.10	7.14	5.31/10.40	5.84/11.03
$\mathcal{A}_m^*$	{1}	4	6.31	7.58	5.54/11.34	6.44/11.56
$\mathcal{A}_m$	{1}	4	5.91	6.82	5.17/10.37	5.48/10.65
$\mathcal{A}_m$	{2}	4	**5.89**	**6.77**	**5.04/10.31**	**5.43/10.60**
$\mathcal{A}_m$	{4}	4	5.93	6.87	5.22/10.56	5.62/10.81
$\mathcal{A}_m$	{10}	4	5.97	6.95	5.24/10.61	5.64/10.82
$\mathcal{A}_m$	{1}	4	**5.91**	**6.82**	**5.17/10.37**	**5.48/10.65**
$\mathcal{A}_m$	{1-3}	4	5.92	6.91	5.29/10.44	5.55/10.75
$\mathcal{A}_m$	{1-6}	4	5.85	6.82	5.21/10.39	5.65/10.79
$\mathcal{A}_m$	{1}	2	6.23	7.34	5.36/11.06	6.01/11.32
$\mathcal{A}_m$	{1}	4	5.91	6.82	5.17/**10.37**	**5.48/10.65**
$\mathcal{A}_m$	{1}	6	5.89	**6.81**	**5.14**/10.41	5.50/10.66
$\mathcal{A}_m$	{1}	8	**5.78**	7.01	5.20/10.43	5.52/10.71

$\mathcal{A}_g$ denotes the proposed gated adapter layer, and $\mathcal{A}_m$ denotes the proposed multi-basis adapter layer. 'POS' denotes the injection position of the adapter layer. '$\#B_k$' denotes #bases, the number of bases in $\mathcal{A}_m$. The '$*$' in $\mathcal{A}_m^*$ denotes the system trained without the proposed predictor regularization. **Bold** indicates the best results for each sub-table.

Adapter Layer Type: Firstly, two different adapter layer types are performed and shown as the top part of Table II. Four bases are used in the multi-basis adapter layer in default, and a dense-based down-up module with inner dimension=64 is used for each basis with both scaling and shifting connection ($\tau_1 = 1$ and $\tau_2 = 1$). It shows that the proposed layer-wise adaptation can significantly improve the system performance on the accent data, while also obtain obvious WER reduction on the accent-unspecific standard Librispeech data. Both two adapter layer types work well, and get the similar accuracy. For the multi-basis adapter layer, to verify the importance of the proposed predictor regularization with multi-task learning, another system without the predictor regularization is also conducted, and the results are shown as $\mathcal{A}_m^*$ in Table II. It is observed that the multi-basis adapter layer cannot work well without the predictor regularization module, and the predictor regularization with multi-task learning is important for the proposed multi-basis adaptation. For better understanding, we print $\boldsymbol{\alpha}$ for observation, and found that the weight almost falls on one basis, e.g. $\boldsymbol{\alpha} \approx (0, 1, 0, 0)$. In



(a) Basis One (b) Basis Two

(c) Basis Three (d) Basis Four

Fig. 5. Boxplot and violinplot visualization of the interpolation weight distributions for each basis in the multi-basis adapter layer. The vertical axis shows the interpolation weight $\boldsymbol{\alpha}_i$, where i is the basis index. The horizontal axis is the accent categories. Here, 4-basis adapter layer is applied, and we plot all data in accent test by boxplot and violinplot to visualize the mean value of each accent.

this case, $\mathcal{A}_m(H_i) \approx H_i$, which means the adapter layer does not work, therefore the system performance is only at the same level with the baseline model.

Adapter Layer Position, Number and Basis Quantity: Different adapter layer positions are firstly explored for both gated and multi-basis adapter layers, and the layer positions $\{1\}, \{2\}, \{4\}, \{10\}$ are chosen.[2] It shows that almost all positions of the end-to-end ASR model can get obvious improvement, and the lower position, i.e. $\{1\}$ or $\{2\}$, is more suitable than the higher position for the proposed layer-wise adaptation architecture. Then multiple adapter layers are compared and it is performed with multi-basis adaptation. $\{1\}, \{1\text{-}3\}, \{1\text{-}6\}$ mean 1, 3, 6 adapter layers respectively.[3] It shows that there is even no more improvement with multiple adapter layers compared to the single adapter layer. Considering multiple adapter layer will increase the model parameters, so we utilize one single adapter layer in the following experiments. Finally the '#bases' number is investigated and the results are shown as the last 4 lines in Table II. It is observed that the system performance seems be improved gradually with more bases in the multi-basis adapter layer, but the improvement is very limited when #bases$\geq$4. To keep compact adaptation parameters, 4-basis is the good choice and also be used in the following experiments.

Visualization on Multi-Basis Adapter Layer: Fig. 5 shows the interpolation weight distributions on each basis from the four-basis adapter layer model. The large interpolation parameter of one basis is considered having more correlation to this accent. It can be clearly seen that different bases capture a different set of highly correlated accents. For example, *Basis Two* focuses mostly on the Portuguese (PT) accent, and then the American (US) and Russian (RU) accents. The inherent correlation shown in Fig. 4 between different accents can be also revealed from this figure. For example, British (U.K.) and American (US) accents have consistently high correlations with *Basis One*, and much lower correlations with other bases. Meanwhile, Indian (IND) and Japanese (JPN) are far from each other, which also have distinct preferences for bases: IND accent prefers *Basis Four*, while JPN accent prefers *Basis Three*. If we further compare this illustration with the accent embedding visualization in Fig. 4, we can find that the correlation and distance within different accents are consistent. These illustrations further demonstrate that the proposed multi-basis adapter layer can well-capture the accent-dependent information with the guidance of accent embeddings, thus finally improve the multi-accent speech recognition.

C. Comparison of Different Adaptation Methods

The best configuration for the proposed layer-wise adaptation can be chosen based on the experiments in the above section. Different adaptation methods are applied and the comparison with detailed results are illustrated in Table III for multi-accent speech recognition. Several traditional methods for multi-accent ASR are also performed, including 'fine-tune,' 'concatenation with i-vector and accent embeddings' and 'MTJR' [53]. 'Fine-tuning' is the intuitive method, which re-trains and fine-tunes the model on the available 160 hours accented data in this work. 'Concat' means we concatenate i-vector (or the accent embedding) with down-sampled Fbank features, and feed it into the model directly. 'MTJR' denotes 'multi-task learning with joint speech and accent recognition,' which is recently proposed in [53] for multi-accent speech recognition.

It shows that although the other methods can obtain WER reduction on the corresponding seen accented data, the improvement is usually limited on the unseen accented data such as Spain (ES) here. So these methods are easy to be implemented, but are not feasible for unseen accent generalization. Moreover, for the accent-unspecific standard data, the performance degrades catastrophically with fine-tuning strategy, which is undesirable in the real implementation. Compared with deep accent embeddings, i-vector adaptation gains much smaller on both accented and accent-unspecific test sets in both intuitive concatenation and the proposed layer-wise adaptation method. The recently proposed multi-task learning with joint speech and accent recognition (MTJR) can slightly improve the accent data, but also degrade obviously on the accent-unspecific standard data, which is not discussed in the original MTJR paper [53].

In contrast, the proposed layer-wise adaptation strategy, no matter using the gated adapter layer $\mathcal{A}_g$ or the multi-basis adapter layer $\mathcal{A}_m$, can get larger, significant and consistent improvements on all seen accent, unseen accent and accent-unspecific standard speech data, which shows the advantages than the other traditional methods. Furthermore, we combine $\mathcal{A}_g$ and $\mathcal{A}_m$ together with one entire architecture by computing the output as:

$$\begin{aligned} \boldsymbol{H}_2^g &\leftarrow \boldsymbol{H}_2 + \mathcal{A}_g(\boldsymbol{H}_2, \boldsymbol{z}), \\ \boldsymbol{H}_2^{g,m} &\leftarrow \boldsymbol{H}_2^g + \mathcal{A}_m(\boldsymbol{H}_2^g, \boldsymbol{z}), \end{aligned} \tag{19}$$

[2]It is worth noting that the encoder transformer layers are numbered starting from 1 to 12, and the adapter layer is then inserted before the i-th encoder layer.

[3]Here $\{m\text{-}n\}$ means injecting adapter layers into the $m^{\text{th}}\sim n^{\text{th}}$ encoder blocks.

TABLE III
PERFORMANCE (WER) (%) COMPARISON OF BASELINE SYSTEM AND DIFFERENT ADAPTATION METHODS

Model	Seen Accent								Unseen Accent		Accent		Libri	
	US	UK	IND	CHN	JPN	PT	RU	KR	CAN	ES	cv	test	dev c/o	test c/o
Baseline	5.75	3.17	9.32	13.49	6.66	6.38	11.04	6.80	5.21	9.88	6.54	7.61	5.64/11.43	6.31/11.68
Fine-tune	4.92	2.82	8.34	12.00	5.92	5.66	9.78	5.82	4.25	9.18	5.85	6.83	7.84/13.03	8.67/13.94
Concat (i-vector)	5.10	2.94	8.91	12.81	6.24	5.92	10.78	6.21	4.84	9.42	6.34	7.52	5.51/11.22	5.81/11.43
Concat (accent)	4.87	2.79	8.83	12.23	6.18	5.84	10.57	6.04	4.38	9.41	6.21	7.21	5.61/11.07	5.72/10.98
MTJR [53]	5.03	2.79	8.54	11.89	6.23	5.82	10.28	6.12	4.13	9.57	6.23	7.09	6.89/12.00	7.71/12.69
$\mathcal{A}_g$(i-vector)	5.42	3.08	9.01	13.41	6.39	6.01	10.84	6.88	5.02	9.39	6.31	7.49	5.30/11.15	5.50/11.25
$\mathcal{A}_g$	5.35	2.63	8.74	11.25	5.84	5.71	9.76	6.02	4.39	8.79	5.87	6.77	5.05/10.22	5.53/10.72
$\mathcal{A}_m$	5.30	**2.57**	8.68	11.46	5.81	5.69	9.66	5.97	4.53	8.68	5.89	6.77	5.04/10.31	5.43/10.60
$\mathcal{A}_g$ + $\mathcal{A}_m$	**4.91**	2.58	**8.48**	**11.15**	**5.70**	**5.59**	**9.51**	**5.71**	**4.28**	**8.38**	**5.66**	**6.71**	**4.61/10.10**	**5.23/10.58**

$\mathcal{A}_g$ denotes the proposed geted adapter layer, and $\mathcal{A}_m$ denotes the proposed multi-basis adapter layer.

TABLE IV
PERFORMANCE (WER) (%) COMPARISON OF THE PROPOSED LAYER-WISE ADAPTATION WITH DEEP ACCENT EMBEDDINGS ON DIFFERENT END-TO-END ASR MODELS

Model	Accent		Libri	
	cv	test	dev c/o	test c/o
RNN-T	8.31	9.03	7.31/15.38	8.24/15.23
+ Layer-Wise Adapt	**7.04**	**7.81**	**6.13/13.61**	**6.90/13.09**
Transformer	6.54	7.61	5.64/11.43	6.31/11.68
+ Layer-Wise Adapt	**5.66**	**6.71**	**4.61/10.10**	**5.23/10.58**
Conformer	6.21	7.21	5.09/10.98	5.84/10.60
+ Layer-Wise Adapt	**5.53**	**6.56**	**4.14/9.51**	**4.08/9.36**

where $\boldsymbol{z}$ is the accent embedding, $\boldsymbol{H}_2$ is the input of 2-nd adapted encoder block or the input of $\mathcal{A}_g$, $\boldsymbol{H}_2^g$ is the output of $\mathcal{A}_g$ as well as the input of $\mathcal{A}_m$, and $\boldsymbol{H}_2^{g,m}$ is the output of $\mathcal{A}_m$ as well as the input of the later original 2-nd encoder block. It is observed that the final system using both $\mathcal{A}_g$ and $\mathcal{A}_m$ adapter layer obtains an additional improvement, and it outperforms the baseline with about 15.0% WER reduction on all seen/unseen accent data and accent-unspecific standard data sets. These results suggest that the proposed methods can learn accent-dependent information effectively and improve the robustness of speech recognition against accent variations.

D. Layer-Wise Adaptation on Different End-to-End ASR

The proposed layer-wise adaptation approach is further applied on different end-to-end ASR models to evaluation the generalization of the method. Besides the above joint CTC/Attention model with Transformer, Conformer and RNN-Transducer are also implemented with the multi-basis adapter layer, and the results are illustrated in Table IV. The results show that the proposed layer-wise adaptation works consistently well on different end-to-end architectures, and significant improvements can be obtained on both accent-specific data and accent-unspecific standard data.

VII. CONCLUSION

In this paper, a layer-wise fast adaptation is proposed for multi-accent ASR in end-to-end model. To get the accurate accent representation, a deep model based deep accent representation learning method is firstly developed, with some useful strategies, including PPG feature, TTS-based data augmentation, test-time augmentation and multi-embedding fusion. With this new approach, high-performance accent classification and accurate accent representations can be obtained.

Assisted with the above accurate accent embeddings, the layer-wise adaptation architecture is explored for fast accent adaptation in ASR, and two types of adapter layers are proposed, which make the accent adaptation flexible and effective. This new accent adaptation approach is significantly better than the traditional methods on multi-accent speech recognition, and works well on different kinds of end-to-end ASR models. More importantly, another advantage of the new adaptation approach is that it can get consistent improvement not only on the seen accent in the training data, but also on the unseen accents. Moreover, it avoids catastrophic performance degradation for fine-tuning method on accent-unspecific standard speech. The system with our proposed method gets consistent $\sim$15.0% WER reduction on all seen/unseen accents and accent-unspecific standard speech data.

REFERENCES

[1] C. Huang, T. Chen, S. Li, E. Chang, and J. Zhou, "Analysis of speaker variability," in *Proc. 7th Eur. Conf. Speech Commun. Technol.*, 2001, pp. 1377–1380.
[2] C. Huang, T. Chen, and E. Chang, "Accent issues in large vocabulary continuous speech recognition," *Int. J. Speech Technol.*, vol. 7, no. 2, pp. 141–153, 2004.
[3] X. Shi et al., "The accented english speech recognition challenge 2020: Open datasets, tracks, baselines, results and methods," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6918–6922.
[4] A. Hinsvark et al., "Accented speech recognition: A survey," 2021, *arXiv:2104.10747*.
[5] N. Hirayama, K. Yoshino, K. Itoyama, S. Mori, and H. G. Okuno, "Automatic speech recognition for mixed dialect utterances by mixing dialect language models," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 2, pp. 373–382, Feb. 2015.
[6] M. Najafian, A. DeMarco, S. Cox, and M. Russell, "Unsupervised model selection for recognition of regional accented speech," in *Proc. Proc. Interspeech*, 2014, pp. 2967–2971.
[7] T. Fukuda et al., "Data augmentation improves recognition of foreign accented speech," in *Proc. Interspeech*, 2018, pp. 2409–2413.
[8] Y. Huang, D. Yu, C. Liu, and Y. Gong, "Multi-accent deep neural network acoustic model with accent-specific top layer using the KLD-regularized model adaptation," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 2977–2981.

[9] S. Ghorbani, S. Khorram, and J. H. Hansen, "Domain expansion in DNN-based acoustic models for robust speech recognition," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, 2019, pp. 107–113.

[10] B. Houston and K. Kirchhoff, "Continual learning for multi-dialect acoustic models," in *Proc. Interspeech*, 2020, pp. 576–580.

[11] X. Yang, K. Audhkhasi, A. Rosenberg, S. Thomas, B. Ramabhadran, and M. Hasegawa-Johnson, "Joint modeling of accents and acoustics for multi-accent speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 1–5.

[12] A. Jain, M. Upreti, and P. Jyothi, "Improved accented speech recognition using accent embeddings and multi-task learning," in *Proc. Interspeech*, 2018, pp. 2454–2458.

[13] A. Jain, V. P. Singh, and S. P. Rath, "A multi-accent acoustic model using mixture of experts for speech recognition," in *Proc. Interspeech*, 2019, pp. 779–783.

[14] B. Li et al., "Multi-dialect speech recognition with a single sequence-to-sequence model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 4749–4753.

[15] K. Rao and H. Sak, "Multi-accent speech recognition with hierarchical grapheme based models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 4815–4819.

[16] H. Hu et al., "REDAT: Accent-invariant representation for end-to-end ASR by domain adversarial training with relabeling," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6408–6412.

[17] S. Yoo, I. Song, and Y. Bengio, "A highly adaptive acoustic model for accurate multi-dialect speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 5716–5720.

[18] A. Das, K. Kumar, and J. Wu, "Multi-dialect speech recognition in english using attention on ensemble of experts," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6244–6248.

[19] M. Grace, M. Bastani, and E. Weinstein, "Occam's adaptation: A comparison of interpolation of bases adaptation methods for multi-dialect acoustic modeling with LSTMs," in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2018, pp. 174–181.

[20] H. Zhu, L. Wang, P. Zhang, and Y. Yan, "Multi-accent adaptation based on gate mechanism," in *Proc. Interspeech*, 2019, pp. 744–748.

[21] X. Gong, Y. Lu, Z. Zhou, and Y. Qian, "Layer-wise fast adaptation for end-to-end multi-accent speech recognition," in *Proc. Interspeech*, 2021, pp. 1274–1278.

[22] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 5329–5333.

[23] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Interspeech*, 2020, pp. 3830–3834.

[24] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 4960–4964.

[25] T. Hori, S. Watanabe, and J. R. Hershey, "Joint CTC/attention decoding for end-to-end speech recognition," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, 2017, pp. 518–529.

[26] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 4835–4839.

[27] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, pp. 369–376.

[28] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu, "Deep feature for text-dependent speaker verification," *Speech Commun.*, vol. 73, pp. 1–13, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[30] H. Zeinali, S. Wang, A. Silnova, P. Matějka, and O. Plchot, "But system description to voxceleb speaker recognition challenge 2019," 2019, *arXiv:1910.12592*.

[31] X. Xiang, S. Wang, H. Huang, Y. Qian, and K. Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," in *Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, 2019, pp. 1652–1656.

[32] S. Wang, J. Rohdin, O. Plchot, L. Burget, K. Yu, and J. Černockỳ, "Investigation of specaugment for deep speaker embedding learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7139–7143.

[33] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4690–4699.

[34] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[35] L. Sun, H. Wang, S. Kang, K. Li, and H. M. Meng, "Personalized, cross-lingual TTS using phonetic posteriorgrams," in *Proc. Interspeech*, 2016, pp. 322–326.

[36] G. Zhao, S. Ding, and R. Gutierrez-Osuna, "Foreign accent conversion by synthesizing speech from phonetic posteriorgrams," in *Interspeech*, 2019, pp. 2843–2847.

[37] C. Du, B. Han, S. Wang, Y. Qian, and K. Yu, "SynAug: Synthesis-based data augmentation for text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 5844–5848.

[38] G. Sun et al., "Generating diverse and natural text-to-speech samples using a quantized fine-grained VAE and autoregressive prosody prior," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 6699–6703.

[39] Y. Ren et al., "Fastspeech: Fast, robust and controllable text to speech," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 3171–3180.

[40] J.-M. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 5891–5895.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 1097–1105, 2012.

[42] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," in *Proc. NeurIPS Deep Learn. Symp.*, 2016. [Online]. Available: https://openreview.net/forum?id=BJLa_ZC9

[43] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst*., 2017, pp. 6000–6010.

[44] A. Gulati et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[45] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 5206–5210.

[46] D. Povey et al., "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2011. [Online]. Available: http://www.danielpovey.com/files/2011_asru_kaldi.pdf

[47] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proc. Conf. Empirical Methods Natural Lang. Process.: Syst. Demonstrations*, 2018, pp. 66–71.

[48] S. Watanabe et al., "ESPNet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

[49] D. S. Park et al., "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[50] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," 2015, *arXiv:1510.08484*.

[51] J. Shen et al., "Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 4779–4783.

[52] H. Huang, X. Xiang, Y. Yang, R. Ma, and Y. Qian, "AIspeech-SJTU accent identification system for the accented english speech recognition challenge," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6254–6258.

[53] J. Zhang, Y. Peng, V. T. Pham, H. Xu, H. Huang, and E. S. Chng, "E2E-based multi-task learning approach to joint speech and accent recognition," in *Proc. Interspeech*, 2021, pp. 1519–1523.

Yanmin Qian (Senior Member, IEEE) received the B.S. degree from the Department of Electronic and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2007, and the Ph.D. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2012. Since 2013, he has been with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, where he is currently a Full Professor. From 2015 to 2016, he was an Associate Research with the Speech Group, Cambridge University Engineering Department, Cambridge, U.K. His research interests include the acoustic and language modeling in speech recognition, speaker and language recognition, speech enhancement and separation, key word spotting, and multimedia signal processing.

Xun Gong received the B.Eng. degree from Zhiyuan College, Shanghai Jiaotong University, Shanghai, China, in 2021. He is currently working towards the Ph.D. degree with the X-LANCE Lab, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, under the supervision of Yanmin Qian. His research interests include speech recognition and its adaptation.

Houjun Huang (Member, IEEE) received the Ph.D. degree from the Key Laboratory of Speech Acoustics and Content Understanding, University of Chinese Academy of Sciences, Beijing, China, in 2017. Since 2017, he has been with the AISpeech Ltd, Suzhou, China, where he is currently an R&d Engineer. His current research interests include speaker and language recognition, key word spotting, and multimedia signal processing.