IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Fine-tuning Self-Supervised Learning Models for End-to-End Pronunciation Scoring

AHMED ZAHRAN, (Member, IEEE), ALY FAHMY, KHALED WASSIF, and HANAA BAYOMI.
Cairo University, Faculty of Computers and Artificial Intelligence, 26H2+4HP, Ad Doqi, Dokki, Giza Governorate 3750010

Corresponding author: Ahmed Zahran (e-mail: zahran@ieee.org).

ABSTRACT Automatic pronunciation assessment models are regularly used in language learning applications. Common methodologies for pronunciation assessment use feature-based approaches, such as the Goodness-of-Pronunciation (GOP) approach, or deep learning speech recognition models to perform speech assessment. With the rise of transformers, pre-trained self-supervised learning (SSL) models have been utilized to extract contextual speech representations, showing improvements in various downstream tasks. In this study, we propose the end-to-end regressor (E2E-R) model for pronunciation scoring. E2E-R is trained using a two-step training process. In the first step, the pre-trained SSL model is fine-tuned on a phoneme recognition task to obtain better representations for pronounced phonemes. In the second step, transfer learning is used to obtain a pronunciation scoring model that uses a Siamese neural network to compare the pronounced phoneme representations to embeddings of the canonical phonemes and produce the final pronunciation scores. E2E-R achieves a Pearson correlation coefficient (PCC) of 0.68, which is similar to the state-of-the-art GOPT-PAII model while eliminating the need for training on additional native speech data, feature engineering, or external forced alignment modules. To our knowledge, this work presents the first utilization of a pre-trained SSL model for end-to-end phoneme-level pronunciation scoring on raw speech waveforms. [a]

[a]The code is available at https://github.com/ai-zahran/E2E-R.

INDEX TERMS automatic pronunciation assessment, pronunciation scoring, pre-trained speech representations, self-supervised speech representation learning, wav2vec 2.0, wavlm, hubert

I. INTRODUCTION

The task of pronunciation assessment is concerned with assessing a language learner's pronunciation of phonemes, words, and sentences in the foreign language they are learning. Amidst the rapid growth of globalization and the increasing demand for learning a second language, the need for automatic pronunciation assessment to be integrated into language learning software is continuously increasing. Two main approaches have been proposed to tackle this problem. The first is the mispronunciation detection and diagnosis (MDD) approach, which aims to detect the phonemes mispronounced by the learner and diagnose the learner's mispronunciation of these phonemes. The second approach is the pronunciation scoring approach. In this approach, the goal is to produce a pronunciation score for the speech produced by the second language learner. This score can be a phoneme-level, word-level, or utterance-level score. In this study, we focus on phoneme-level pronunciation scoring.

A. RELATED WORK

The history of related work in MDD and pronunciation scoring can be partitioned into three phases. In the first phase, feature-based techniques were used along with classical Hidden Markov Model-Gaussian Mixture Model (HMM-GMM)-based speech recognition techniques. The second phase started after the utilization of deep neural networks (DNNs) in speech recognition. In this phase, DNNs were utilized for recognizing phonemes and extracting speech representations for phoneme classification and scoring. After the introduction of self-supervised learning (SSL) approaches into speech research, a third phase began where SSL approaches and pre-trained SSL models were utilized

for pronunciation assessment.

1) Feature-Based Approaches

Previous efforts in pronunciation scoring utilized feature-based techniques. These methods usually used forced alignment in order to obtain phoneme boundaries in an utterance and extract features for each phoneme based on its time segment. The extracted features would then be fed to another model that was trained on the pronunciation scoring task in order to score the speaker's pronunciation of the phoneme. In [1], phoneme log-likelihood scores from a Hidden Markov Model (HMM) speech recognition system were used to score phonemes pronunciations. This work was later enhanced by using phoneme posterior scores [2], which obtained relatively better results. In [3], the authors introduced the Goodness of Pronunciation (GOP) score, which was calculated by normalizing the log-phoneme posterior probabilities through time.

2) Deep Neural Network-Based Approaches

After the advent of deep learning, multiple models based on deep neural networks (DNNs) were introduced for pronunciation scoring. In [4], a transformer encoder is utilized to encode audio input, and another transformer encoder is used to encode text input. The outputs from both encoders are then fed to a multi-head attention layer [5], using the text representations as queries and audio representations as keys and values. A multi-layer perceptron (MLP) is used to obtain word-level and sentence-level scores. Recently, [6] employed transfer learning and attention on deep features directly from the DNN-HMM-based acoustic model for pronunciation scoring. The attention mechanism is applied to obtain a weighted representation of a word's phonemes, then these weighted representations are averaged to obtain word-level representations, which are also weighted by an attention mechanism and averaged to obtain utterance-level representations. A non-linear transformation and a sigmoid function are applied to the utterance-level representations to obtain utterance-level pronunciation scores. Additionally, multiple studies have investigated the use of deep learning techniques to enhance the GOP method. In [7], the authors compute a pronunciation score using a DNN-HMM model through a new formula that utilizes HMM senone posteriors and state transition probabilities. In [8], a new model called context-aware GOP (CaGOP) is introduced. CaGOP uses self-attention to compute duration factors, which are used along with transition factors to calculate the CaGOP score.

3) Self-Supervised Learning-Based Approaches

With the recent introduction of the transformer architecture and self-supervised learning (SSL) in speech recognition [9], [10], multiple studies have been presented to utilize transformers and pre-trained SSL models in phoneme recognition and pronunciation assessment. English et al. perform probing over the self-supervised representations extracted from an SSL model to obtain phonetic features [11]. In [12], the authors experiment with a standard transformer and the wav2vec 2.0 architecture [10] for the tasks of mispronunciation detection and diagnosis. In [13], an SSL model is fine-tuned over non-native speech using connectionist-temporal classification (CTC). The fine-tuned model is used to extract contextual representations, which are used along with the corresponding text to produce a pronunciation score using a Bi-LSTM. Chao et al. introduced 3M [14], which employs embeddings from multiple pre-trained SSL models in addition to other speech features. 3M utilizes the transformer encoder architecture to provide the final phoneme, word, and utterance-level scores.

In this study, we aim to utilize a pre-trained SSL model for building an end-to-end pronunciation scorer. We use transfer learning to build a phoneme recognition model using the SSL model. This model is then used to obtain phoneme representation for the pronounced phonemes using an attention-based decoder. The phoneme representations obtained through the model are then used to produce pronunciation scores for each phoneme by comparing these representations to the embeddings of the corresponding canonical phonemes. This comparison is carried out using a Siamese neural network [15], which takes the phoneme representations and the corresponding phoneme embeddings and produces phoneme pronunciation scores. The canonical phoneme embeddings to which we compare the pronounced phoneme representations are produced from an embedding layer that is trained during the scorer's training process.

The produced model can be considered end-to-end in terms of joint modeling [16], as all components are joined in a single computational graph, from the feature extraction phase up to the scoring phase. This eliminates the need for feature engineering to obtain the audio representations that are used as input to the scoring algorithm and enables the optimization of all the components of the model simultaneously from the self-supervised model that takes the raw waveform as input to the similarity scoring module that produces the final score. Furthermore, the model does not use an external forced-alignment algorithm to obtain start and end timestamps for the phonemes, since the decoder is forced to produce one representation per canonical phoneme.

The study is organized as follows: In section II, we discuss some related background work. Section III introduces the proposed method. We discuss the process used to train a phoneme recognition system using transfer learning over an SSL model and the various components used therein. We also discuss the Siamese neural network used for scoring. In section IV, we discuss the experimental setup, provide information about the corpora and the training parameters we used, and present the results of our experiments. The study is concluded with a summary and plans for future studies in section V.

II. BACKGROUND WORK

In this section, we discuss background work about the SSL models we used in our experiments, namely wav2vec 2.0,

[10], HuBERT [17], and WavLM [18].

A. WAV2VEC 2.0

wav2vec 2.0 [10] uses the transformer architecture to produce contextualized representations for a speech signal. The model first starts by producing latent representations from a speech waveform using a convolutional encoder. This convolutional encoder consists of a sequence of convolutional layers that are applied directly to the speech signal. The function of the convolutional encoder $f : X \rightarrow Z$ is to map a sequence of speech samples $\boldsymbol{x} = (x_1, x_2, ..., x_N)$, to a sequence of latent speech representations $\boldsymbol{Z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_T)$, where x_i is the speech sample at time step i, N is the total number of speech samples, $\boldsymbol{z}_j$ is the j-th latent speech representation produced by the encoder, and T is the length of the latent speech representations sequence produced by the encoder. The latent speech representations produced by the encoder are used to learn contextualized representations through a contrastive learning task [19].

In order to learn contextualized representations, spans of the latent representations are masked, and then the representations are fed into a transformer network. The transformer network $g : Z \rightarrow C$ uses self-attention to map the latent representations $\boldsymbol{Z}$ into contextualized representations $\boldsymbol{C} = (\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_T)$, where $\boldsymbol{c}_i$ is the contextualized representation produced at time step i. The model then solves a contrastive learning task over the masked spans, in which it tries to pick the correct latent representation corresponding to the contextualized representation from a list of latent representations. The list of latent representation candidates consists of the correct candidate and a set of noisy examples from other masked spans in the same utterance.

In the contrastive learning task, the target speech representations are first quantized into discrete units. The model learns this discretization through Gumbel-Softmax [20]. This process is done jointly while learning the contextualized representations. The authors argue that this approach is better than learning each process individually. The loss function is given by the equation:

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d \tag{1}$$

where $\mathcal{L}_m$ is the contrastive loss function, and $\mathcal{L}_d$ is a codebook diversity loss that penalizes the model when it doesn't use the codebook entries equally. Given $K + 1$ candidate representations $\widetilde{\boldsymbol{q}} \sim \boldsymbol{Q_t}$ to choose from, with K noisy samples and 1 correct sample $\boldsymbol{q}_t$, the contrastive loss function for a contextualized representation $\boldsymbol{c}_t$ is given by the equation:

$$\mathcal{L}_m = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{c}_t, \boldsymbol{q}_t)/\kappa)}{\sum_{\widetilde{\boldsymbol{q}} \sim \boldsymbol{Q}_t} \exp(\mathrm{sim}(\boldsymbol{c}_t, \widetilde{\boldsymbol{q}})/\kappa)} \tag{2}$$

where $\mathrm{sim}(.,.)$ is the cosine similarity function, and κ is the contrastive temperature term. The authors set this term to 0.1.

B. HUBERT

The HuBERT [17] architecture resembles the wav2vec 2.0 architecture in its main components. Both include a convolutional encoder that extracts latent representation followed by a transformer that produced contextualized representations. However, the self-supervised training task in HuBERT differs from wav2vec 2.0.

In wav2vec 2.0, the task is to obtain contextualized representations that achieve high cosine similarity with the corresponding latent representations in order to be able to distinguish these corresponding latent representations from noisy samples. In HuBERT, latent representations are assigned to clusters by a clustering algorithm, and the task is to obtain contextualized representations that can be used to accurately predict the clusters to which the corresponding latent representations are assigned. A projection layer is added on top of the transformer to predict cluster assignments. In the first iteration, this clustering process is performed on Mel-frequency cepstrum coefficient (MFCC) features. In the following iterations, the clustering process is performed on latent representations extracted by the model from the previous iteration. In order to obtain classification targets with different granularities, the authors use cluster ensembles with different codebook sizes.

Given a sequence $\boldsymbol{X}$ of length T and a set of masked indices $M \subset [T]$, the corrupted sequence $\tilde{\boldsymbol{X}}$ is generated by replacing $\boldsymbol{x}_t$ by a mask embedding $\tilde{\boldsymbol{x}}$ for $t \in M$. For the target sequence $\boldsymbol{Z}^{(k)}$ from the k-th clustering model, the cross-entropy loss function is defined as:

$$L_m(f; \boldsymbol{X}, \{\boldsymbol{Z}^{(k)}\}_k, M) = \sum_{t \in M} \sum_k \log p_f^{(k)}(z_t^{(k)} | \tilde{\boldsymbol{X}}, t) \tag{3}$$

where $p_f(.|\tilde{\boldsymbol{X}}, t)$ is a distribution over target clusters for time step t predicted by model f.

To calculate the distribution over target clusters, the transformer outputs are compared to codeword embeddings computed through an embedding layer. For a masked time step at time t, the corresponding transformer output $\boldsymbol{o}_t$ is multiplied by a projection matrix $\boldsymbol{A}^{(k)}$ to obtain a representation that can be compared to the codeword embedding. The distribution is parametrized with

$$p_f^{(k)}(c|\tilde{\boldsymbol{X}}, t) = \frac{\exp(\mathrm{sim}(\boldsymbol{A}^{(k)}\boldsymbol{o}_t, \boldsymbol{e}_c)/\tau)}{\sum_{c'=1}^{C} \exp(\mathrm{sim}(\boldsymbol{A}^{(k)}\boldsymbol{o}_t, \boldsymbol{e}_{c'})/\tau)} \tag{4}$$

where $\boldsymbol{e}_c$ is the embedding for codeword c, $\mathrm{sim}(.,.)$ is the cosine similarity function, and τ is used for scaling the logit. The authors set τ to 0.1. The loss function is only computed over masked time steps. The authors argue that this forces the model to learn the long-range temporal structure of speech data since the model faces masked timestamps and therefore has to depend on the context surrounding these masked timestamps to predict their cluster assignments.

To obtain an ASR model from the resulting pre-trained model, the CNN encoder of the pre-trained model is frozen, and the projection layer is replaced with a softmax layer. The

resulting model is then trained on an ASR task by optimizing the model's parameters over the CTC loss function.

C. WAVLM

The WavLM [18] model is inspired by the architecture of HuBERT. WavLM aims to produce an SSL model that can be used for full-stack speech processing tasks. The architecture consists of a CNN encoder and a transformer model similar to HuBERT and wav2vec 2.0. The task is to predict cluster assignments of masked subsequences similar to HuBERT.

WavLM mainly differs from HuBERT in the data it uses for pre-training and in its implementation of gated relative position bias in the transformer [21]. In WavLM's pre-training process, a larger and more diverse training set is used when compared to HuBERT. This makes the model more resilient to changes in the environment and to background noises. Furthermore, the data is corrupted by simulating noise and overlapped speech. This forces the model to implicitly learn denoising and speech separation, which is shown by its superior performance on different speech tasks from the SUPERB evaluation benchmark [22] and multiple other benchmarks, including benchmarks for speaker verification, speech separation, and speaker diarization.

III. PROPOSED ARCHITECTURE

An overview of the proposed scorer architecture is illustrated in Figure 1. The scorer architecture consists of two modules: the pronounced phoneme representation module and the Siamese scoring module. Given a speech utterance and its corresponding canonical pronunciation, the pronounced phoneme representation module is responsible for extracting a representation for each phoneme the user is expected to pronounce according to the canonical pronunciation, while the Siamese scoring module compares the representations produced by the pronounced phoneme representation module to the embeddings of the corresponding canonical phonemes.

The pronounced phoneme representation module consists of a pre-trained SSL model, an encoder-decoder network, and a phoneme embedding layer that is used by the decoder. The SSL model's purpose is to extract acoustic features from raw audio, while the encoder's purpose is to encode the acoustic features that were extracted by the SSL model, and the decoder's task is to produce a phoneme representation for each of the pronounced phonemes given the output of the encoder. The components of the first module are fine-tuned jointly on a phoneme recognition task using the joint CTC-attention loss [23] before being utilized for the phoneme scoring task.

The Siamese scoring module includes a Siamese neural network [15] that compares the pronounced phonemes with the canonical phonemes, in addition to an embedding layer that is responsible for obtaining embeddings for each of the canonical phonemes to be used for comparison. The representation of the pronounced phoneme that is produced by the pronounced phoneme representation module is passed into one branch of the Siamese neural network, while the embedding of the corresponding canonical phoneme is fed as input to the other branch of the network. A cosine similarity function [24] is used to calculate the similarity score for the outputs from the two branches of the Siamese neural network. The objective function we attempt to minimize is the mean squared error between the obtained similarity scores and the human-annotated pronunciation scores for each phoneme.

In the rest of this section, we discuss the automatic phoneme recognition (APR) model we build in the first stage and its training process using joint CTC-attention. We also discuss the pronunciation scoring model we build in the second stage and the Siamese neural network that it uses. We start by explaining the architecture of the encoder and the decoder used in our model. We follow this with a discussion of how we train the phoneme recognition model which contains the SSL model and the encoder-decoder network using the joint CTC-attention training mechanism. Finally, we discuss the pronunciation scoring stage, where we examine the canonical label forcing method used to obtain phoneme representations from the decoder, the Siamese neural network used in the scoring module, and the embedding layers used in the pronounced phoneme representation and Siamese scoring modules.

A. ENCODER-DECODER NETWORK

Given an input sequence $\boldsymbol{x} = (x_1, ..., x_{T_x})$, where T_x is the length of the input sequence, and a phoneme label sequence $\boldsymbol{y} = (y_1, ..., y_{T_y})$, where T_y is the length of the phoneme label sequence, the goal of the encoder is to encode the features extracted by the SSL model from the input sequence $\boldsymbol{x}$ into a sequence of hidden representations, while the goal of the decoder is to decode those hidden representations into a sequence of phoneme representations.

1) Encoder

The encoder we use is a vanilla neural network which consists of a number of linear layers followed by an activation function. In order to obtain acoustic features, WavLM [18] is used as the pre-trained SSL model to extract high-level speech features from the input sequence $\boldsymbol{x}$. The output of the pre-trained model is fed into the encoder, which outputs a set of hidden representations $\boldsymbol{H} = (\boldsymbol{h}_1, ..., \boldsymbol{h}_{T_H})$, where T_H is the length of the hidden representations sequence produced by the encoder. These hidden representations are then processed by a linear transformation followed by a softmax activation to produce a set of output probabilities for each frame that will be used by the CTC loss function described in III-B1. In addition, the hidden representations are fed to the decoder described in the following subsection.

2) Decoder

An attentional recurrent neural network (RNN) decoder is utilized for decoding the representations created by the encoder and obtaining the values that represent the pronunciations to be scored. Given the input sequence $\boldsymbol{x}$, the decoder

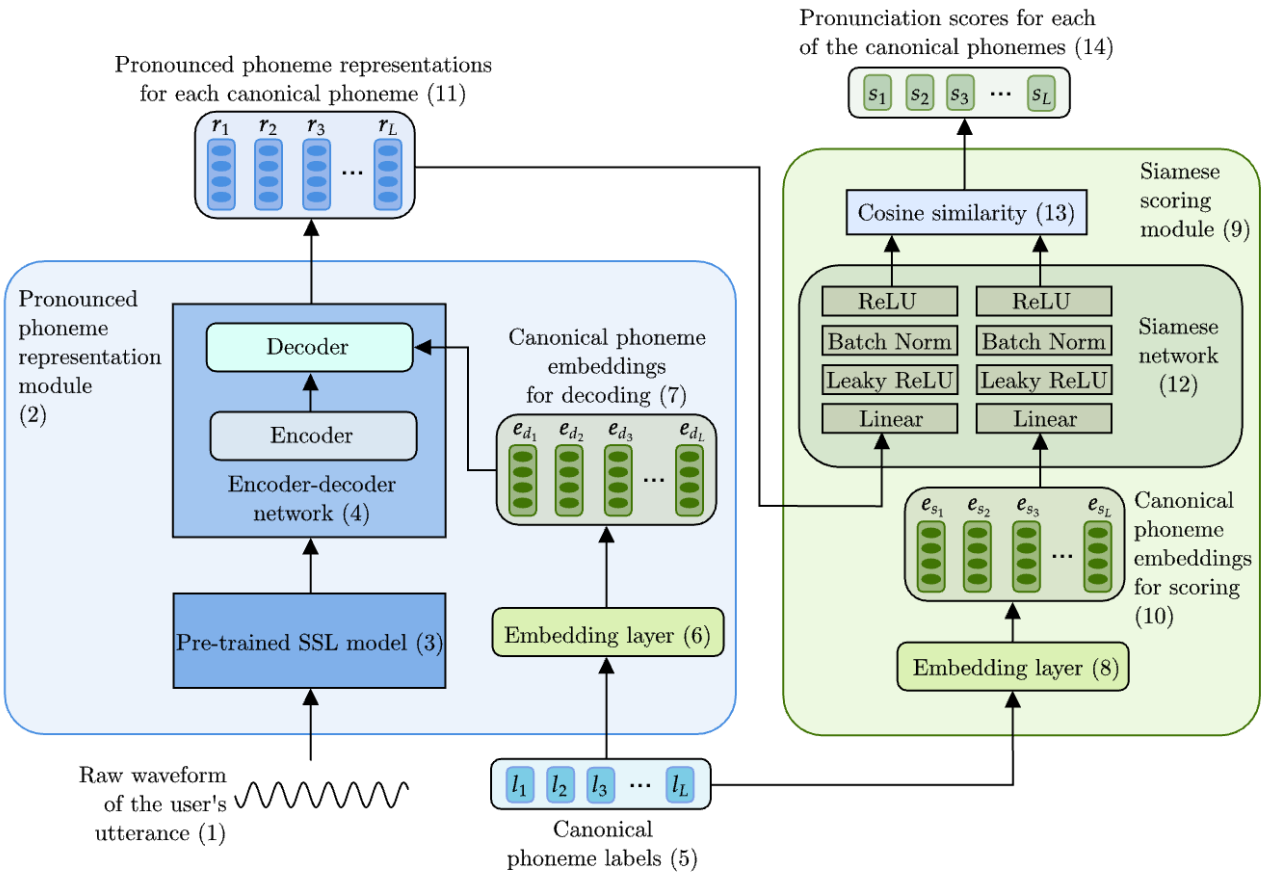


FIGURE 1. The proposed E2E-R scorer architecture. The raw audio waveform (1) is first fed into the pronounced phoneme representation module (2) in order to extract pronounced phoneme representations. The first component of this module is the pre-trained SSL model (3), which extracts speech features from the waveform and feeds them to the encoder-decoder network (4). The sequence of phoneme labels of the canonical pronunciation (l_1, ... l_L) (5) is fed to two embedding layers. The first embedding layer (6), which is part of the pronounced phoneme representation module, produces phoneme embeddings ($\boldsymbol{e_{d_1}}$, .., $\boldsymbol{e_{d_L}}$) (7) to be used in the decoding process. The second embedding layer (8), which is part of the Siamese scoring module (9), produces the canonical phoneme embeddings ($\boldsymbol{e_{s_1}}$, ..., $\boldsymbol{e_{s_L}}$) (10) to be used during the scoring phase. In the encoder-decoder network (4), the encoder is used to encode the high-level speech features produced by the pre-trained model (3), while the decoder takes the encodings produced by the encoder along with the phoneme embeddings (7) produced by the first embedding layer (6) and produces pronounced phoneme representations ($\boldsymbol{r_1}$, ..., $\boldsymbol{r_L}$) which correspond to the canonical phonemes. The pronounced phoneme representations (11) and the extracted canonical phoneme embeddings (10) from the second embedding layer (8) are both fed to a Siamese neural network (12) which consists of one linear layer followed by a leaky ReLU activation and batch normalization. The final output is passed into a ReLU activation. The outputs of the Siamese neural network are compared using Cosine similarity (13) to produce the final pronunciation scores (14).

produces a sequence of phoneme representations that can be used to predict the output sequence $\boldsymbol{y}$, The prediction of the output sequence given the input from the encoder is done according to the recursive equation:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{T_y} P(y_i|\boldsymbol{H}, y_{1:i-1}) \tag{5}$$

The training process thus consists of feeding the whole sequence of hidden representations produced by the encoder to the decoder, and for each time step the target symbol from the previous time step is used as input to the decoder along with the hidden representations.

The conditional probability in equation 5, can be defined as:

$$P(y_i|\boldsymbol{H}, y_{1:i-1}) = g(y_{i-1}, \boldsymbol{s}_i, \boldsymbol{c}_i) \tag{6}$$

where g is a non-linear function, $\boldsymbol{s}_i$ is the decoder state at time step i, and $\boldsymbol{c}_i$ is the context vector for the target phoneme at time step i. The decoder state $\boldsymbol{s}_i$ is computed as:

$$\boldsymbol{s}_i = f(\boldsymbol{s}_{i-1}, y_{i-1}, \boldsymbol{c}_i) \tag{7}$$

where f is a recurrence function and $\boldsymbol{c}_i$ is the context vector. During training, we use the ground truth labels as input to the decoder in equation 7 while predicting the next time step, which is referred to as teacher forcing [25]. In order to do this, the start-of-sentence token is prepended to the symbol sequence to serve as input for the first time step of the decoding process.

The context vector $\boldsymbol{c}_i$ is calculated as a weighted sum of the hidden representations $\boldsymbol{H}$:

$$\boldsymbol{c}_i = \sum_{j=1}^{T_H} \alpha_{i,j} \boldsymbol{h}_j \tag{8}$$

where $\alpha_{i,j}$ is the alignment between the output symbol y_i and the annotation $\boldsymbol{h_j}$. The alignments for a time step i are calculated using an attention function. In this study, we use a location-aware attention function [26].

To compute the recurrence function f, we use a gated recurrent unit (GRU) neural network [27]. The cell output of the GRU network for each time step i is concatenated to the context vector of this time step $\boldsymbol{c}_i$ calculated in equation 8 and fed to a linear projection to obtain the time step's decoder output. This results in a sequence of decoder outputs $\boldsymbol{R} = (\boldsymbol{r}_1, ..., \boldsymbol{r}_{T_y})$, where $\boldsymbol{r}_i$ is the decoder output corresponding to the target symbol y_i. Another linear transformation is applied to this decoder output, followed by a softmax activation to obtain $P(y_i|\boldsymbol{x}, y_{1:i-1})$ in equation 6.

B. PHONEME RECOGNITION TRAINING

The training process starts by training an APR model that utilizes the pre-trained SSL model. The input to the APR model is an audio waveform of an utterance, and the expected output is the sequence of phonemes uttered in this utterance. The joint CTC-attention architecture is utilized for phoneme recognition training. This architecture has been previously utilized in numerous studies for phoneme recognition and mispronunciation detection [28]–[31].

1) Connectionist Temporal Classification (CTC) Loss

The goal of CTC is to train the model to maximize the marginal probability distribution $P(\boldsymbol{y}|\boldsymbol{x})$ over all possible alignment sequences $\Phi(\boldsymbol{y'})$:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{\pi} \in \Phi(\boldsymbol{y'})} P(\boldsymbol{\pi}|\boldsymbol{x}) \tag{9}$$

where $\boldsymbol{y'}$ is equal to the sequence $\boldsymbol{y}$ after inserting the blank symbol ϵ in the start, in the end, and between every two consecutive symbols, and $\Phi(\boldsymbol{y'})$ is a function that derives all possible alignment sequences $\boldsymbol{\pi} = (\pi_1, ..., \pi_{T_x})$ of the label sequence over $\boldsymbol{x}$. The alignment sequences allow consecutive repetitions of the same phoneme. Using this formulation for the probability distribution $P(\boldsymbol{y}|\boldsymbol{x})$, the CTC loss to be minimized can be defined as:

$$\mathcal{L}_{CTC} = -\ln P(\boldsymbol{y}|\boldsymbol{x}) \tag{10}$$

In order to compute the term $P(\boldsymbol{\pi}|\boldsymbol{x})$, we assume conditional independence between labels. This term can then be approximated as:

$$P(\boldsymbol{\pi}|\boldsymbol{x}) \approx \prod_{t=1}^{T_x} P(\pi_t|\boldsymbol{x}) = \prod_{t=1}^{T_x} q_t(\pi_t) \tag{11}$$

where $q_t(\pi_t)$ represents the model's prediction probability for the symbol π_t at time t. To compute this formula efficiently, the CTC algorithm utilizes a dynamic programming approach to compute the forward and backward variables which represent the total probability of the u-the label at time t:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \sum_{u=1}^{|\boldsymbol{y'}|} \frac{\alpha_t(u)\beta_t(u)}{q_t(y'_u)} \tag{12}$$

where $\alpha_t(u)$ is the forward variable, and $\beta_t(u)$ is the backward variable. Taking the derivative of this equation with respect to output probability calculated using q_t produces the gradient which can be used to perform backpropagation.

2) Attention Loss

The attention loss is computed as the mean negative log-likelihood (NLL) loss over the ground truth label posteriors obtained from the decoder.

$$\begin{aligned}\mathcal{L}_{Attention} &= -\ln P(\boldsymbol{y}|\boldsymbol{x}) \\ &= -\sum_{i=1}^{T_y} \ln P(y_i|\boldsymbol{H}, y_{1:i-1})\end{aligned} \tag{13}$$

The decoding process should stop once the end-of-sentence symbol is emitted. Therefore, the ground truth label sequence used in equation 13 is modified by appending an end-of-sentence label.

3) Joint CTC-Attention Training

Joint CTC-attention training is performed by minimizing a loss function that includes both CTC and attention losses. This loss function is given by the equation:

$$\mathcal{L}_{CTC-Attention} = \lambda \mathcal{L}_{CTC} + (1-\lambda)\mathcal{L}_{Attention} \tag{14}$$

where λ is a tunable parameter, $0 \leq \lambda \leq 1$.

C. PRONUNCIATION SCORING TRAINING

After training the APR model, transfer learning is used to train a pronunciation scoring model that utilizes the fine-tuned SSL model and the encoder-decoder network. To obtain one output from the decoder per canonical phoneme, we use a method of canonical label forcing, which we discuss in subsection III-C1. This results in the sequence of outputs from the decoder $\boldsymbol{R}$, which are used as a sequence of pronounced phoneme representations. We compare these representations to the canonical pronunciations of the corresponding phonemes to obtain the pronunciation scores.

1) Canonical label forcing

In automatic speech recognition (ASR) and APR tasks where the goal is to generate the sequence of labels for the input, the decoder utilizes the output of each time step of the decoding process in the next step during inference. This is performed using any of the well-known decoding algorithms like greedy search and beam search [16]. However, during training, we use teacher forcing to force the decoder to stay as close as possible to the ground truth labels.

In the phoneme pronunciation scoring problem, since the canonical phoneme sequence is provided for each utterance and provided the assumption that the input utterance represents the user's attempt to pronounce this phoneme sequence, we exploit the nature of the decoder's input by using the canonical labels as an approximation for ground truth labels during inference in a manner that is similar to the way teacher

forcing is used during APR training. This process forces the decoder to produce one output representation per canonical phoneme and guides the decoding process by forcing it to stay close to the expected labels. Namely, for each time step, we feed the canonical pronunciation label of the previous time step to the decoder instead of its predicted output of the previous time step.

Therefore, given the canonical phoneme sequence $\boldsymbol{y}$, we prepend the beginning-of-sentence symbol to the canonical phoneme sequence. Then in each decoding time step, we feed the label for the previous time step as input to the decoder instead of its predicted output from the previous time step while computing the decoder state $\boldsymbol{s}_i$ in equation 7. The decoder output produced for the final time step should correspond to the end-of-sentence symbol and is therefore discarded from the decoder outputs that are later utilized as pronounced phoneme representations.

2) Siamese neural network

In order to compare the user's pronunciation of a phoneme with the canonical pronunciation of this phoneme, a Siamese neural network is used to compare the representation obtained from the decoder with the embedding vector of the canonical phoneme. Given the canonical phoneme sequence $\boldsymbol{y}$, the canonical phoneme embedding vector $\boldsymbol{u}_i$ is calculated for each canonical phoneme y_i in the sequence. This embedding is compared to the pronounced phoneme representation $\boldsymbol{r}_i$ obtained from the decoder to produce the similarity score. The similarity score for the canonical embedding vector and the pronounced phoneme representation is computed as their cosine similarity:

$$v_i' = \mathrm{sim}(t(\boldsymbol{u}_i), t(\boldsymbol{r}_i)) \tag{15}$$

where $\mathrm{sim}(.,.)$ is the cosine similarity function and t is a transformation followed by a ReLU activation. The ReLU activation is used to produce non-negative output, which enables us to obtain a non-negative cosine similarity that can be used as the predicted pronunciation score. Given the corresponding human-annotated score v_i, the loss is calculated as the mean squared error between the computed scores and the human-annotated scores:

$$\mathrm{loss} = \frac{1}{T_l}\sum_{j=1}^{T_l}(v_i - v_i')^2 \tag{16}$$

3) Embedding layers

We use two embedding layers in our architecture to feed phoneme embeddings to our model. The first embedding layer is responsible for creating phoneme embeddings that are used as input to the decoder. These embeddings are necessary during the APR training phase, where the decoder uses the embedding for each time step to predict the output of the next time step. These same embeddings are also used in the scoring phase for canonical label forcing as described in subsection III-C1. The second embedding layer produces the canonical phoneme embeddings that are fed to the Siamese neural network in the scoring phase in order to be used as one side of the comparison in the scoring process, with the pronounced phoneme representations serving as the other side of the comparison.

Both embedding layers use a weight matrix $\boldsymbol{W} \in \mathbb{R}^{|\mathbb{V}|\times d_e}$, where $\mathbb{V}$ is the vocabulary of symbols and d_e is the embedding dimension. The vocabulary includes the phoneme set being used plus three more symbols which represent the start-of-sequence symbol, the end-of-sequence symbol, and the blank symbol. The requested embedding $\boldsymbol{w}_x$ for a symbol x is retrieved using the formula:

$$\boldsymbol{w}_{\boldsymbol{x}} = \boldsymbol{W}\boldsymbol{x}' \tag{17}$$

where $\boldsymbol{x}'$ is the one-hot vector for symbol x.

IV. EXPERIMENTS

A. CORPORA

Two corpora are employed in this study. The first corpus, TIMIT [32], is a speech corpus with phoneme-level transcriptions, which we use for training an automatic phoneme recognition model. The second corpus, speechocean762 [33], is a pronunciation scoring corpus that contains manually annotated pronunciation scores. This corpus is used for training the pronunciation scoring model.

1) TIMIT

The TIMIT corpus consists of audio utterances from native English speakers along with the word-level and phoneme-level annotations for these utterances. The corpus includes utterances from 630 speakers of 8 American English dialects. Each dialect is represented by one male and one female speaker, and each speaker recorded 10 utterances. The audio recordings for the utterances are 16-bit with a 16 kHz sampling rate. The corpus is further divided into training, development, and test sets.

2) speechocean762

The speechocean762 corpus consists of 5,000 utterances spoken by 250 non-native English speakers. Both the ratios of adults-to-children and males-to-females are 1:1. The mother tongue of the speakers is Mandarin. Each speaker used his/her phone to record 20 prompted texts, yielding a total of 6 hours of recorded speech. The corpus also includes manual annotations for each utterance. These annotations include phoneme-level, word-level, and utterance-level scores. In this study, we focus on phoneme-level scores. The annotators give a score to each phoneme that ranges from 0 to 2, with 2 indicating correct pronunciation, 1 indicating a heavy accent pronunciation, and 0 indicating incorrect pronunciation. The phoneme-level scores are scaled to a scale from 0 to 1 during the training process by dividing each score by the maximum score value. In our case, the maximum score value is 2. Before computing the metrics, the output is rescaled back to a scale from 0 to 2 by multiplying the output by the maximum score value.

B. TRAINING PROCEDURE

The training scripts used in this study were built using the SpeechBrain toolkit [34]. The training process was performed on a V100 GPU. The experiments were carried out using an architecture that consists of a self-supervised learning model, followed by an encoder-decoder network with the purpose of extracting one speech representation vector for each expected phoneme in the canonical pronunciation. The model was trained on 16 kHz audio and therefore expects an input of the same sampling rate. The corpora use a phoneme dictionary that consists of 39 phonemes. Adding the beginning-of-sentence symbol, the end-of-sentence symbol, the blank symbol, and the silence symbol, we end up with a dictionary of 43 symbols.

The training process consists of two stages. In the first stage, an APR model that consists of the SSL model and the encoder-decoder network described in III-A is trained on the TIMIT corpus in the manner described in III-B. In the second stage, the fine-tuned SSL model, the encoder, the decoder, and the decoder's phoneme embedding layer are kept, while the linear transformations and softmax activations that were used to obtain phoneme transcription from the outputs of the encoder and the decoder are removed. The scoring module which consists of the canonical phoneme embedding layer and the Siamese neural network described in III-C is added to this architecture to produce the final model architecture. This architecture is then trained on the speechocean762 dataset to produce the final model.

The WavLM-large model [1] pre-trained on 60k hours of Libri-Light [35], 10k hours of GigaSpeech [36], and 24k hours of VoxPopuli [37] is used as the SSL model for feature extraction. The output of this model is normalized using layer normalization and fed into the encoder. The encoder is a Vanilla neural network of two layers, each consisting of 1,024 neurons, followed by a Leaky ReLU activation. A linear projection and a softmax activation are applied to the output of the encoder to produce the output to be used for calculating the CTC loss. The number of neurons in this linear projection is equal to the total number of symbols in the dictionary.

The decoder employed in this study is an attentional RNN decoder. An embedding layer is used to obtain a 128-element embedding vector for each phoneme. This decoder receives the high-level speech representations obtained from the encoder, along with the corresponding canonical phoneme embeddings during the training phase. The decoder consists of one GRU layer with attention followed by a linear projection. The best results were obtained by setting the number of neurons in the GRU layer to 256. A study of the different options we explored for the GRU layer size is discussed in section IV-D2. The recurrent weights are initialized using orthogonal initialization, and Xavier initialization is used to initialize input connection weights. The attention utilized in the decoder is location-aware attention [26] with 10 channels,

[1] https://huggingface.co/microsoft/wavlm-large

TABLE 1. E2E-R model parameters.

Parameter	Value
SSL Model	
Learning rate	0.0001
new-bob annealing factor	0.9
SpecAugment	
Perturbation speeds	95, 100, 105
Encoder	
Input size	1,024
Number of layers	2
Number of neurons per layer	1,024
Decoder Phoneme Embedding Layer	
Dictionary size	43
Embedding size	128
Canonical Phoneme Embedding Layer	
Dictionary size	43
Embedding size	256
Decoder	
Number of layers	1
Number of neurons	256
Number of attention neurons	256
Attention distribution scaling factor	1.0
Number of location-aware attention channels	10
Location-aware attention kernel size	100
Dropout factor	0.5
CTC Loss Linear Projection	
Input size	1,024
Number of neurons	43
Attention Loss Linear Projection	
Input size	256
Number of neurons	43
Beam Search	
Beam size	16
Siamese neural Network	
Input size	256
Number of layers	1
Number of neurons	1,024
Other Parameters	
Batch size	4
Learning rate	0.0003
new-bob improvement threshold	0.0025
new-bob annealing factor	0.8
CTC weight	0.2
Sampling Rate	16 kHz
Sequential loss label smoothing proportion	0.1

a kernel size of 100, and a distribution scaling factor of 1.0. The number of internal and output attention neurons is 256. An additional 0.5-factor dropout is applied. Similar to the encoder, a linear projection and a softmax activation are applied to the output of the decoder to produce the output to be used for calculating the attention loss. The number of neurons in this linear projection is also equal to the total number of symbols in the dictionary. Label smoothing of 0.1 is applied to the attention loss.

During the APR training phase in the first stage, the extracted embeddings for each time step are fed to the decoder in order to predict the output for the next time step. During the validation and testing phases, greedy search and beam search algorithms are used, respectively, to obtain the predicted output for each time step, which is similarly used as input to the decoder to predict the output for the next time step. For beam search, the beam size is set to 16.

In the second stage, we obtain embeddings for each of the canonical phonemes using the embedding layer in the Siamese scoring module. The size of the embedding vector is equal to the number of neurons of the decoder. The embedding vector obtained for each canonical phoneme is then compared with the corresponding speech representation vector that is extracted for the pronounced phoneme by the pronounced phoneme representation module. This comparison is carried out using the Siamese neural network. The Siamese neural network consists of a single linear layer of 1,024 neurons, followed by a leaky ReLU activation, a batch normalization, and a final ReLU activation for obtaining a non-negative output. The similarity score is computed for the resulting vectors corresponding to canonical phonemes and pronounced phonemes using Cosine similarity.

A time-domain approximation of SpecAugment [38] is used for data augmentation. The data augmentation algorithms perform three types of data augmentation: dropping audio chunks, dropping frequency bands, and speed perturbation with 95, 100, and 105 speeds. Training is performed for 20 epochs in the first stage and for 100 epochs in the second stage, with the batch size set to 4. Two optimizers using the ADAM optimization algorithm [39] are used during the training process. The first optimizer is used for the WavLM network, and the second optimizer is used for the rest of the architecture. The training process starts with a learning rate of 0.0001 for the WavLM optimizer, and 0.0003 for the other optimizer. The model is optimized on a joint CTC-attention loss function, where the CTC loss weight is set to 0.2, and the attention loss weight is set to 0.8. During the training phase for pronunciation scoring, a third optimizer is added with an initial learning rate of 0.0003. The objective function used for the pronunciation scoring training phase is the mean squared difference between the predicted scores and the actual scores. During the training process, learning rate annealing with new-bob scheduler [2] is performed to modify the learning rates of each of the optimizers. The new-bob scheduler annealing factor is set to 0.9 and 0.8 for the SSL model and for the rest of the parts of the architecture, respectively. The improvement threshold for the scheduler is set to 0.0025. Table 1 summarizes the parameters used in the E2E-R model.

[2]Originally mentioned in https://www1.icsi.berkeley.edu/Speech/faq/nn-train.html

C. BASELINE MODELS

In this study, we compare our results to four baseline models using the speechocean762 dataset [33] as a benchmark. One of these baselines is the original model published with the speechocean762 dataset. The other three are to the best of our knowledge the current state-of-the-art models in phoneme scoring. The baseline models we mention in this study can be classified into two classes based on the way the evaluation metrics were reported in their publications:

- Rounded score baselines: For these baselines, the reported evaluation metrics are based on the pronunciation scores produced by these models after rounding to the nearest integer. This class includes the original speechocean762 Kaldi Recipe (GOP-based Feature).
- Unrounded score baselines: The reported evaluation metrics for these baselines are based on the pronunciation scores without rounding. This class includes GOPT (Librispeech and PAII-A) [40] and 3M [14].

The rest of this section discusses the technical details of these baselines.

1) speechocean762 Kaldi Recipe (GOP-based Feature)

This method was proposed by Zhang et al. [33] and was implemented in the gop_speechocean762 S5 Kaldi recipe, which is included with the Kaldi toolkit [41]. The proposed model is a neural network-based goodness of pronunciation (NN-GOP) [42] scoring method. A time-delay neural network (TDNN) is pre-trained on the utterances produced by native speakers from the LibriSpeech dataset [43]. The pre-trained TDNN is used to perform forced alignment and obtain GOP-based features, which are then used to train a regression model for each phoneme for the task of pronunciation scoring.

2) GOPT (Librispeech)

The GOPT model [40] proposed by Gong et al. is a transformer model that performs multi-granularity pronunciation scoring. The proposed method starts by performing forced alignment for canonical phonemes using frame-level posteriors that are produced by an acoustic model. This is done in order to enable the extraction of GOP features. The transformer then takes phoneme representations for each canonical phoneme as input. These representations are the result of the addition of projected GOP features, canonical phoneme embeddings, and positional embeddings. In addition, five $[cls]$ tokens are prepended to the sequence of input phoneme representations. These $[cls]$ tokens correspond to utterance accuracy, fluency, completeness, prosodic, and total

scores. The transformer model produces outputs for each of the phonemes and the $[cls]$ tokens. Regression heads for phoneme-level and word-level scores are added on top of the produced outputs accordingly to produce phoneme-level and word-level scores. Utterance-level regression heads are added on top of the five utterance-level outputs corresponding to the five $[cls]$ tokens to calculate the utterance-level scores.

The GOPT (Librispeech) baseline is a version of the model that uses an acoustic model that was trained on the Librispeech dataset. The acoustic model is a factorized time-delay neural network (TDNN-F) acoustic model that is trained on the 960-hour dataset [3]. This acoustic model is used in GOPT for forced alignment and feature extraction.

3) GOPT (PAII-A)

The GOPT (PAII-A) [40] is a variation of the GOPT model that uses a TDNN-F acoustic model that is trained on a combination of open-source English data and internal L2 data from native speakers collected by PAII. The model shows improved performance on the pronunciation scoring task and demonstrates the importance of native speech training data for the GOPT architecture.

4) 3M

3M [14] is an enhancement over GOPT. The authors augment the feature set previously used for GOPT with prosodic features and features extracted using SSL models. A projection layer is used to project these acoustic features into the feature space that is expected by the transformer. The model also uses vowel/consonant positional embeddings, which are projected to the feature space that the model expects and added to the features along with the phoneme embeddings and the positional embeddings in a manner that is similar to the GOPT feature computation.

D. RESULTS

The work in this study is compared against the baseline models discussed in IV-C based on two evaluation metrics: mean squared error (MSE) and Pearson correlation coefficient (PCC) [44]. The formulas for calculating MSE and PCC for a subset of the data (train, dev, or test set) are shown in equations 18 and 19, respectively.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(v_i - v_i')^2 \tag{18}$$

$$\text{PCC} = \frac{\sum_{i=1}^{N}(v_i - \overline{v})(v_i' - \overline{v'})}{\sqrt{\sum_{i=1}^{N}(v_i - \overline{v})^2 \sum_{i=1}^{N}(v_i' - \overline{v'})^2}} \tag{19}$$

where v_i is the human-annotated pronunciation score for a phoneme, v_i' is the predicted pronunciation score for the same phoneme, $\overline{v}$ is the mean human annotation score for all phonemes in the subset, $\overline{v'}$ is the mean predicted score for all phonemes in the subset, and N is the total number of phonemes in the subset.

In this study, we choose the best model in terms of PCC, as we find that PCC reflects the task of measuring the performance of the pronunciation scoring system most accurately. Therefore, other models that were produced in our experiments may not be selected as the best model despite achieving lower MSE than the best model. Figure 2 shows the training and validation losses per epoch for our best model. Figures 3 and 4 show the MSE and PCC achieved by the best model per epoch on the validation set. Although we run the training process for 100 epochs, it is clear from the plots that by epoch 30 the loss and the MSE have almost saturated, and by epoch 50 the PCC has also saturated. The rest of this section compares the results of our model with the results of the baseline models and presents the ablation study of our model.

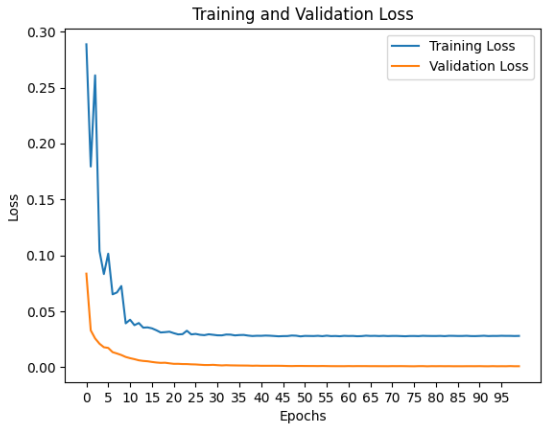


FIGURE 2. Losses per epoch on the training and validation sets.

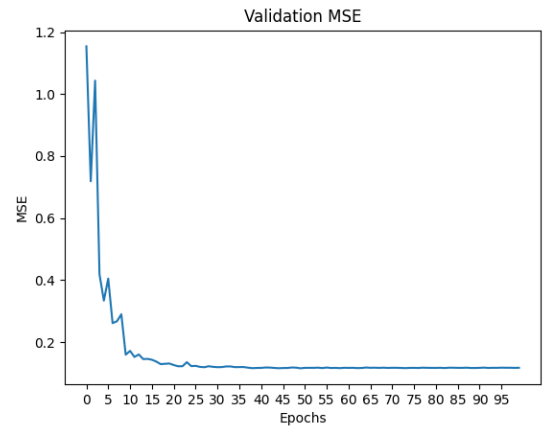


FIGURE 3. MSE per epoch on the validation set.

[3] https://kaldi-asr.org/models/m13

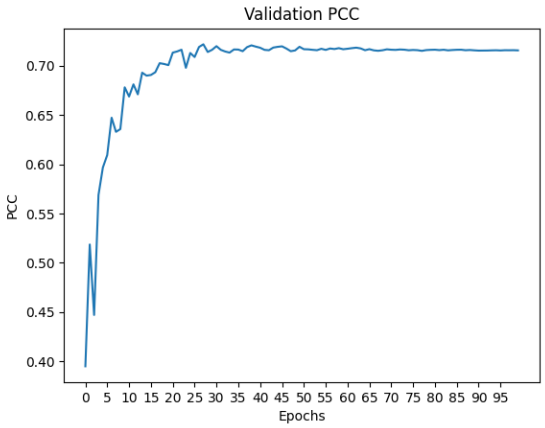


FIGURE 4. PCC per epoch on the validation set.

TABLE 2. Comparison of MSE and PCC results for E2E-R with the baseline models on the speechocean762 test set.

Model	MSE (↓)	PCC (↑)	Rounded MSE (↓)	Rounded PCC (↑)
GOP-based feature [33]	-	-	0.16	0.45
GOPT (Librispeech) [40]	0.085	0.612	-	-
GOPT (PAII-A) [40]	**0.069**	**0.679**	-	-
3M [14]	0.078	0.656	-	-
E2E-R	0.085	**0.678**	**0.1134**	**0.5931**

TABLE 3. Ablation study results for E2E-R on the speechocean762 test set. The name of the best model is bolded.

Model	MSE (↓)	PCC (↑)	Rounded MSE (↓)	Rounded PCC (↑)
Pre-trained SSL Model				
WavLM	0.085	**0.678**	**0.113**	**0.593**
HuBERT	**0.083**	0.662	0.114	0.573
wav2vec 2.0	0.088	0.652	0.121	0.557
Phoneme Representation Module				
APR Encoder-Decoder	**0.085**	**0.678**	**0.113**	**0.593**
LSTM	0.306	0.552	0.398	0.441
Decoder Size				
128	0.082	0.668	0.111	0.580
256	0.085	**0.678**	0.113	**0.593**
512	0.080	0.668	0.111	0.582
1,024	**0.079**	0.676	**0.109**	**0.593**
Loss Function				
MAE	0.091	0.645	0.121	0.579
MSE	**0.085**	**0.678**	**0.113**	**0.593**
Similarity Metric				
NSES	0.240	0.621	0.342	0.281
Cosine Similarity	**0.085**	**0.678**	**0.113**	**0.593**
Fine-tuning				
APR Training	0.085	**0.678**	0.1134	**0.5931**
APR Training + Fine-tuning on native data	**0.0806**	0.6742	**0.1107**	0.5861

1) Comparison with Baselines

The results for our experiments on the end-to-end regressor (E2E-R) proposed in this paper and the baselines are reported in table 2. As shown in the table, the E2E-R model achieves a 10.78% relative improvement in PCC with respect to GOPT (Librispeech). Moreover, E2E-R achieves a 3.35% relative improvement with respect to 3M, while utilizing only one pre-trained SSL model and no additional speech features, as opposed to 3M's utilization of outputs from multiple pre-trained SSL models along with prosodic features. This shows the advantage of the two-stage training process where the SSL model is fine-tuned jointly with the rest of the architecture.

When it comes to MSE, E2E-R manages to achieve an MSE that is similar to GOPT (Librispeech). However, the 3M model achieves lower MSE than E2E-R. The GOPT (PAII-A) model outperforms E2E-R in both metrics. We assume that this is due to the amount of native L2 data that the acoustic model of GOPT (PAII-A) is trained on, which enables it to extract more accurate speech representations. Nonetheless, E2E-R manages to obtain a PCC that is almost similar to GOPT (PAII-A) without utilizing any additional native speech data for training as opposed to GOPT (PAII-A). Compared to the GOP-based feature model, E2E-R achieves a 29.13% relative decrease in MSE and a 31.80% relative increase in PCC on rounded scores.

2) Ablation Study

In our ablation study, we examine six modifications and their effects on the performance of the system in terms of MSE and PCC. These modifications include experimenting with different pre-trained models, changing the module that is responsible for forming phoneme representations from the original APR encoder-decoder to an LSTM layer, modifying the decoder size, changing the loss function, using a different similarity metric, and fine-tuning the model on native speech data. The results of the ablation study are shown in table 3.

a: Pre-trained SSL Model

The first modification concerns the pre-trained SSL model used for extracting audio features. We experiment with the three most commonly used SSL models, which are WavLM, HuBERT, and wav2vec 2.0. We specifically compare the WavLM-large model with Wav2Vec2-Large-LV60 [4] and Hubert-Large-Finetuned [5]. We notice that the three models give superior performance, with WavLM obtaining the best PCC and HuBERT obtaining the best MSE.

[4]https://huggingface.co/facebook/wav2vec2-large-lv60
[5]https://huggingface.co/facebook/hubert-large-ls960-ft

b: Phoneme Representation Module

In the second modification, we replace the encoder-decoder architecture in the pre-trained APR model with a simple LSTM layer. The aim of this experiment is to examine the effectiveness of the encoder-decoder network of the APR in extracting representative audio representations for each phoneme. We utilize an external acoustic model for forced alignment. In order to be consistent with the baseline models, we use the same TDNN-F model trained on Librispeech as the GOP-based feature and the GOPT (Librispeech) baselines. Using the forced alignment time stamps, we obtain the SSL model-extracted features corresponding to each phoneme and feed these features to a 2-layer LSTM. The LSTM output corresponding to the last SSL model-extracted feature vector for each phoneme is taken as the phoneme representation. The same similarity module is used to compute the similarity between the extracted phoneme representations and the phoneme embeddings. The experiment shows the superior performance of the encoder-decoder model which achieves a lower MSE and a higher PCC. Since the encoder-decoder network is jointly trained on the APR task with the SSL model, this performance is expected.

c: Decoder Size

The third modification is concerned with the size of the decoder. Four experiments were carried out, in which the number of neurons for the decoder is set to 128, 256, 512, and 1,024, respectively. The input size of the sequential layer following the decoder in phoneme recognition training and the number of neurons in the Siamese scoring module's canonical phoneme embedding layer are adjusted accordingly. The best MSE is achieved by setting the number of neurons in the decoder to 1,024 neurons, while the best PCC is achieved by setting the number to 256 neurons.

d: Loss Function

For the fourth modification, we change the loss function to mean absolute error (MAE). MAE is generally less sensitive to outliers since their value is not squared. However, since the range of scores is confined to a specific scale that cannot be exceeded, it is apparent that the edge that MAE has when it comes to handling outliers is not relevant in our case. Therefore, we notice that using MSE achieves higher PCC and lower MSE.

e: Similarity Metric

In our fifth modification, we experiment with the normalized squared Euclidean similarity (NSES) as the similarity metric for pronunciation scores instead of cosine similarity. NSES is the opposite of normalized squared Euclidean distance (NSED) [45]. NSED is defined as the squared distance between two vectors after scaling the lengths of these vectors to have unit norms. For two vectors $\boldsymbol{v}$ and $\boldsymbol{v'}$, NSED can be calculated using the formula:

$$\text{NSED} = 0.5 \cdot \frac{||(\boldsymbol{v} - \overline{\boldsymbol{v}}) - (\boldsymbol{v'} - \overline{\boldsymbol{v'}})||^2}{||(\boldsymbol{v} - \overline{\boldsymbol{v}})||^2 + ||(\boldsymbol{v'} - \overline{\boldsymbol{v'}})||^2} \tag{20}$$

where $\overline{\boldsymbol{v}}$ is the mean of $\boldsymbol{v}$ and $\overline{\boldsymbol{v'}}$ is the mean of $\boldsymbol{v'}$. This can be further simplified as:

$$\text{NSED} = 0.5 \cdot \frac{\text{var}(\boldsymbol{v} - \boldsymbol{v'})}{\text{var}(\boldsymbol{v}) + \text{var}(\boldsymbol{v'})} \tag{21}$$

where $\text{var}(.)$ denotes the variance function. NSED is always in the range of 0 to 1. NSES can therefore be calculated using the formula:

$$\text{NSES} = 1 - \text{NSED} \tag{22}$$

Using NSES as a similarity metric, we noticed a significant decrease in performance reflected in a lower PCC and a significantly higher MSE.

f: Fine-tuning

The sixth and final modification is fine-tuning the APR model on native L2 speech data. In this experiment, we select the correctly-pronounced utterances from the speechocean762 dataset. These utterances are English utterances by non-native speakers in which all the phonemes are pronounced accurately. After performing the initial APR training, we fine-tune the APR model on these selected utterances in order to ingest knowledge of non-native speech into the model. We notice that the scoring model trained on this fine-tuned model achieves a lower MSE. However, the PCC obtained by this model is also reduced.

V. CONCLUSION

In this study, we presented E2E-R, an end-to-end regression model for phoneme pronunciation scoring. An SSL model was utilized for scoring non-native English speakers' pronunciation of English phonemes through a two-step process that utilizes transfer learning. The SSL model was first used to build an automatic phoneme recognition model. This phoneme recognition model was utilized for pronunciation scoring by feeding the phoneme representations extracted by the decoder of this model into a Siamese neural network in order to compare them with canonical phoneme embeddings and obtain a pronunciation score for each phoneme. In future studies, we plan to experiment with different attention and scoring mechanisms and enhance the proposed architecture to produce multi-granularity scores that include word-level and sentence-level scores.

REFERENCES

[1] L. Neumeyer, H. Franco, M. Weintraub, and P. Price, "Automatic text-independent pronunciation scoring of foreign language student speech," in Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96, vol. 3, pp. 1457–1460, IEEE, 1996.

[2] H. Franco, L. Neumeyer, Y. Kim, and O. Ronen, "Automatic pronunciation scoring for language instruction," in 1997 IEEE international conference on acoustics, speech, and signal processing, vol. 2, pp. 1471–1474, IEEE, 1997.

[3] S. M. Witt, S. J. Young, et al., "Language learning based on non-native speech recognition.," in Eurospeech, Citeseer, 1997.

[4] B. Lin and L. Wang, "Attention-based multi-encoder automatic pronunciation assessment," in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7743–7747, IEEE, 2021.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, pp. 5998–6008, 2017.

[6] B. Lin and L. Wang, "Deep feature transfer learning for automatic pronunciation assessment.," in Interspeech, pp. 4438–4442, 2021.

[7] S. Sudhakara, M. K. Ramanathi, C. Yarra, and P. K. Ghosh, "An improved goodness of pronunciation (gop) measure for pronunciation evaluation with dnn-hmm system considering hmm transition probabilities.," in INTERSPEECH, pp. 954–958, 2019.

[8] J. Shi, N. Huo, and Q. Jin, "Context-aware goodness of pronunciation for computer-assisted pronunciation training," arXiv preprint arXiv:2008.08647, 2020.

[9] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," arXiv preprint arXiv:1910.05453, 2019.

[10] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," arXiv preprint arXiv:2006.11477, 2020.

[11] P. C. English, J. Kelleher, and J. Carson-Berndsen, "Domain-informed probing of wav2vec 2.0 embeddings for phonetic features," in Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pp. 83–91, 2022.

[12] M. Wu, K. Li, W.-K. Leung, and H. Meng, "Transformer based end-to-end mispronunciation detection and diagnosis," Proc. Interspeech 2021, pp. 3954–3958, 2021.

[13] E. Kim, J.-J. Jeon, H. Seo, and H. Kim, "Automatic pronunciation assessment using self-supervised speech representation learning," arXiv preprint arXiv:2204.03863, 2022.

[14] F.-A. Chao, T.-H. Lo, T.-I. Wu, Y.-T. Sung, and B. Chen, "3m: An effective multi-view, multi-granularity, and multi-aspect modeling approach to english pronunciation assessment," in 2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), pp. 575–582, IEEE, 2022.

[15] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," Advances in neural information processing systems, vol. 6, 1993.

[16] R. Prabhavalkar, T. Hori, T. N. Sainath, R. Schlüter, and S. Watanabe, "End-to-end speech recognition: A survey," arXiv preprint arXiv:2303.03329, 2023.

[17] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," arXiv preprint arXiv:2106.07447, 2021.

[18] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, et al., "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," IEEE Journal of Selected Topics in Signal Processing, vol. 16, no. 6, pp. 1505–1518, 2022.

[19] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," arXiv preprint arXiv:1807.03748, 2018.

[20] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," arXiv preprint arXiv:1611.01144, 2016.

[21] Z. Chi, S. Huang, L. Dong, S. Ma, B. Zheng, S. Singhal, P. Bajaj, X. Song, X.-L. Mao, H. Huang, et al., "Xlm-e: Cross-lingual language model pre-training via electra," arXiv preprint arXiv:2106.16138, 2021.

[22] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, et al., "Superb: Speech processing universal performance benchmark," arXiv preprint arXiv:2105.01051, 2021.

[23] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp. 4835–4839, IEEE, 2017.

[24] A. Singhal et al., "Modern information retrieval: A brief overview," IEEE Data Eng. Bull., vol. 24, no. 4, pp. 35–43, 2001.

[25] A. M. Lamb, A. G. ALIAS PARTH GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in Advances in Neural Information Processing Systems (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[26] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," Advances in neural information processing systems, vol. 28, 2015.

[27] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," arXiv preprint arXiv:1409.1259, 2014.

[28] S.-W. F. Jiang, B.-C. Yan, T.-H. Lo, F.-A. Chao, and B. Chen, "Towards robust mispronunciation detection and diagnosis for l2 english learners with accent-modulating methods," in 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 1065–1070, IEEE, 2021.

[29] H. Ji, T. Patel, and O. Scharenborg, "Predicting within and across language phoneme recognition performance of self-supervised learning speech pre-trained models," arXiv preprint arXiv:2206.12489, 2022.

[30] B.-C. Yan and B. Chen, "End-to-end mispronunciation detection and diagnosis from raw waveforms," in 2021 29th European Signal Processing Conference (EUSIPCO), pp. 61–65, IEEE, 2021.

[31] L. Zhang, Z. Zhao, C. Ma, L. Shan, H. Sun, L. Jiang, S. Deng, and C. Gao, "End-to-end automatic pronunciation error detection based on improved hybrid ctc/attention architecture," Sensors, vol. 20, no. 7, p. 1809, 2020.

[32] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," NASA STI/Recon technical report n, vol. 93, p. 27403, 1993.

[33] J. Zhang, Z. Zhang, Y. Wang, Z. Yan, Q. Song, Y. Huang, K. Li, D. Povey, and Y. Wang, "speechocean762: An open-source non-native english speech corpus for pronunciation assessment," arXiv preprint arXiv:2104.01378, 2021.

[34] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "SpeechBrain: A general-purpose speech toolkit," 2021. arXiv:2106.04624.

[35] J. Kahn, M. Riviere, W. Zheng, E. Kharitonov, Q. Xu, P.-E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen, et al., "Libri-light: A benchmark for asr with limited or no supervision," in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7669–7673, IEEE, 2020.

[36] G. Chen, S. Chai, G. Wang, J. Du, W.-Q. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang, et al., "Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio," arXiv preprint arXiv:2106.06909, 2021.

[37] C. Wang, M. Riviere, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," arXiv preprint arXiv:2101.00390, 2021.

[38] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," arXiv preprint arXiv:1904.08779, 2019.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[40] Y. Gong, Z. Chen, I.-H. Chu, P. Chang, and J. Glass, "Transformer-based multi-aspect multi-granularity non-native english speaker pronunciation assessment," in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7262–7266, IEEE, 2022.

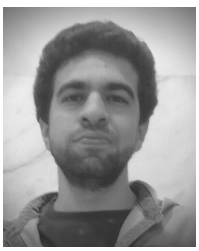
[41] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, IEEE Signal Processing Society, Dec. 2011. IEEE Catalog No.: CFP11SRW-USB.

[42] W. Hu, Y. Qian, F. K. Soong, and Y. Wang, "Improved mispronunciation detection with deep neural network trained acoustic models and transfer learning based logistic regression classifiers," Speech Communication, vol. 67, pp. 154–166, 2015.

[43] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp. 5206–5210, IEEE, 2015.

[44] D. C. Montgomery and G. C. Runger, Applied statistics and probability for engineers. John wiley & sons, 2010.

[45] W. Research, "NormalizedSquaredEuclideanDistance." https://reference.wolfram.com/language/ref/NormalizedSquaredEuclideanDistance.html, 2010. Accessed: 09-June-2023.



AHMED I. ZAHRAN received the B.Sc degree from the Computer Department, Faculty of Engineering, Cairo University, Egypt in 2016.

From 2016 to 2017, he worked in RDI, Egypt as a Machine Learning Researcher, where he worked on Arabic speech recognition in the Hafss project. He joined Cairo University as a Research Assistant from 2017 to 2018, where he worked on a collaborative research project with DataPlus.me. The research project focussed on speech emotion recognition in customer service phone calls. He worked as a Data Scientist in DataPlus.me from 2019 to 2020. In 2020, he joined Agolo (Cairo Office) as a Natural Language Processing Engineer. His research interests include speech recognition, computer-aided language learning, and natural language processing.

ALY A. FAHMY received the B.Sc. degree (Excellent with Honor Grade) in computer engineering from the Computer Department, Military Technical College (M.T.C), in 1972, the DPL - Diploma: General Purpose Simulation from the Computer Department, M.T.C, in 1973, the M.Sc. degree in logical database systems from the Computer Department, E.N.S.A.E, Toulouse, France, in 1976, and the Ph.D. degree in artificial intelligence control of automatic deductions for logic-based systems from the Computer Department, Centre of Research and Studies, C.E.R.T, Toulouse, France, in 1979.

He was the Ex-Dean of the Faculty of Computers and Information, Cairo University. Currently, he is the Dean of the Artificial Intelligence College affiliated to Arab Academy for Science and Technology. He was the Director of the first Center of Excellence in Egypt in the field of Data Mining and Computer Modeling (DMCM) in the period of 2005-2010. DMCM was a virtual research center with more than 40 researchers from universities and industry. With his prominent colleagues, he established the Center for Arabic Language Technologies (ALTEC) in order to bring the Arabic to the international level in man–machine interface in its broader sense.

His research interests include Medical Machine Learning, Data and Text Mining, Computational Linguistics, Text Understanding and Automatic Essay Scoring and Technologies of Man- Machine Interface in Arabic.

KHALID T. WASSIF received his B.Sc. in Accounting with an honorary degree from Faculty of Commerce, Cairo University in 1983. He received his Post G. Diploma in Computer and Information Sciences from Institute of Statistical Studies & Research, Cairo University in 1986. He received his Master and Ph.D. degrees from Cairo University in the field of Artificial Intelligence in 1991 and 1998 respectively.

Dr. Wassif is currently a Professor in Faculty of Computers and Artificial Intelligence, Cairo University. His main research topics include Machine learning, Data mining, Web mining, Case-based reasoning, Big data, and Knowledge engineering. He has supervised or co-supervised twelve students on their Ph.D. dissertations and M.S. theses. He has published thirty research papers in international journals and conference proceedings. He is a reviewer in the international Egyptian Informatics Journal and the Egyptian Computer Journal.

HANAA BAYOMI received the B.S., M.Sc., and Ph.D. degrees (Hons.) in the Faculty of Computer and Artificial Intelligent (computer and information Prev.), Cairo University, Egypt, in 2002, 2006, and 2015, respectively.

She is currently an Associate Professor with the Faculty of Computers and Artificial Intelligence, Cairo University. Her research interests include machine learning, deep learning, Natural language processing, Software Engineering, and soft computing.