RealPRNet: A Real-Time Phoneme-Recognized Network for "Believable" Speech Animation

Zixiao Yu, Haohong Wang, and Jian Ren, *Senior Member, IEEE*

***Abstract*—With the technology development, more and more Internet of Things (IoT) devices with displays are making "face-to-face" interaction through visualization a reality. To protect the privacy of users, communications can be represented through avatars and use audio-driven real-time speech animation. However, if audio is the only available input, the quality of the outcome relies heavily on real-time phoneme recognition, such as recognition accuracy and latency. This article introduces a novel deep-learning-based real-time phoneme recognition network (RealPRNet) scheme to leverage spatial and temporal patterns in the input audio data. With featured long short-term memory stack block and long short-term features, RealPRNet can achieve super performance in phoneme recognition. Our comprehensive empirical results show that compared to the state-of-the-art algorithms, RealPRNet can achieve 20% phoneme error rate (PER) improvement and 4% block error distance (BDE) improvement in the best case.**

***Index Terms*—Deep-learning, Internet of Things (IoT), phoneme recognition, real-time speech animation.**

I. Introduction

THE RAPIDLY developing Internet of Things (IoT) technology is transforming our daily lifestyle. We are also relying more and more on visual interactions and communications since images can convey more complex and diverse information than text and voice. In application scenarios such as unmanned supermarkets, customer service can provide appropriate services to guests from thousands of kilometers away through interaction devices. When the users need help, even though automated question and answer bots already exist that can handle most of the questions, the "face-to-face" communications may provide a far better experience [1], [2] for the customer than the audio-only customer service. While providing face-to-face communications experience, virtual avatar technology also preserves the privacy and the participant parties. In this kind of virtual conversation, the mouth movement of the virtual avatar is an essential module. Human beings are very sensitive to any facial artifacts, uncoordinated or unsynchronized performance of virtual characters, which make facial animation, particularly speech animation production, very challenging since animation involves simultaneous audio recognition and mouth movement animation.

Realistic speech animation [3], [4] has been always very compelling since it can provide the most immersive experiences with high-fidelity human-like virtual characters. However, the high cost involved in the production process and the substantial data requirements, including audio and video, are likely to create privacy issues.

For applications that accept lower realism effects but require good privacy preservation, such as avatar-based online conferences, anonymous alerts, and customer service, audio data could become the only media available during the process. In such scenarios, the virtual characters are required to mimic mouth movement matching the voice input seamlessly, and this is what is called believable speech animation. The "believable" speech animation requires that the algorithm works, under practical resource constraints, for all possible virtual faces with various 3-D models and produces the synthesized video with sufficient realism for users on the remote end to feel comfortable.

The widespread use of NLP technologies, such as speech recognition [5], [6] and speaker identification [7], demonstrates that such a "believable" speech animation is achievable by using only audio input [8]–[10]. In general, the audio input is fragmented into small pieces called frames from where features are extracted. Phoneme, which is the distinct unit of sound, is predicted by using the extracted features and then mapped into the corresponding static mouth shape, called viseme (its counterpart in the visual domain [11]). In such an audio-driven speech animation framework, the accuracy of the phoneme recognition can directly affect the quality of the speech animation.

With the latest advances in deep learning, the phoneme recognition topic has been revisited using completely new methodologies [12]. The advances in phoneme recognition accuracy can lead to better speech animation. Compared to the traditional models, such as the hidden Markov models (HMMs), deep-learning neural network (DNN)-based approaches generally decrease the phoneme recognition error rate by 10%.

For real-time applications, finding the balance between latency and accuracy is critical, as indicated in the design philosophy of RNN and LSTM. The temporal correlation between neighbor audio frames can play a very significant role in a recognition accuracy improvement. However, the phoneme recognition accuracy improvement achieved by adding more neighbor frames in the sliding window is at the cost of latency.

Manuscript received May 7, 2021; revised July 13, 2021; accepted August 22, 2021. Date of publication September 6, 2021; date of current version March 24, 2022. This work was supported in part by NSF under Grant CCF-1919154 and Grant ECCS-1923409. *(Corresponding author: Zixiao Yu.)*

Zixiao Yu and Jian Ren are with the Department of ECE, Michigan State University, East Lansing, MI 48824 USA (e-mail: yuzixiao@msu.edu; renjian@msu.edu).

Haohong Wang is with the Multimedia Lab, TCL Research America, San Jose, CA 95110 USA (e-mail: haohong.wang@tcl.com).

Digital Object Identifier 10.1109/JIOT.2021.3110468

Therefore, a crucial problem that needs to be addressed in order to improve the real-time speech animation quality is to find a phoneme recognition solution that can achieve the best accuracy with reasonably low latency for real-time animation. In this article, we propose a novel deep neural network scheme, called the real-time phoneme recognition network (RealPRNet). With a carefully designed network architecture that considers both temporal and spatial correlations, RealPRNet predicts phoneme stream for a sliding window of audio frames input. A novel concept, called the LSTM stack block (LSB), is introduced to maximize the learning efficiency of the temporal–spatial patterns (more details are covered in the network design section).

To build an end-to-end audio-driven real-time believable speech animation system, the RealPRNet is inserted into the typical speech animation process that converts the audio input into a recognized phoneme sequence and drives a facial animation module. Inspired by the JALI model [10] and properties of the blend shape facial model, the animation is achieved by mapping the recognized phoneme label to a set of parameters to control four basic blend shapes that have hidden physical correlation.

The major contributions of this article can be summarized as follows.

1) We propose a novel neural network-based real-time phoneme recognition scheme (RealPRNet).
2) We develop a real-time audio-driven 3-D speech animation production system.
3) We conduct a comprehensive evaluation to show that the proposed RealPRNet scheme can achieve great improvement over the state-of-the-art algorithms.

The remainder of this article is organized as follows: in Section II, we describe the details of our animation production system components and introduce the deep-learning-based RealPRNet with our insight in Section III. We present an evaluation and experimental results in Section IV. In Section V, we conclude our work and discuss possible directions for future works.

II. Related Work

A typical audio-driven speech animation uses viseme as an intermediary. It can be seen as a combination of phoneme recognition and phoneme-driven speech animation.

A. Phoneme Recognition

Phoneme recognition using audio's fundamental frequency [8] and HMM [13]–[17] has been an active research topic for decades. However, the accuracy of these traditional schemes is not sufficient for speech recognition and speech animation production. The advances in deep learning and neural networks have greatly enhanced the accuracy of phoneme recognition [12], [18]–[20]. In [18], a feedforward deep neural network model was proposed and achieved an error rate of 23.71%. In [12], it was reported that a feedforward deep neural network architecture lowered the error rate to 16.49%. In [20], a network architecture called CLDNN that combines the convolutional neural network (CNN) LSTM and fully connected DNN was proposed. It can further improve the performance for 4%–6% compared to the LSTM. In particular, around 39 different phonemes can be recognized compared to the fundamental schemes [8], which can recognize less than ten different phonemes. However, these methods are all designed for offline situations without latency constraints, which enable any features extraction schemes to be used. In this article, we develop the RealPRNet to achieve accurate real-time phoneme recognition.

B. Speech Animation

In [8], the audio input is fragmented into small pieces, called frames, where fundamental frequency features are extracted from. Phonemes are predicted by recognizing the vowels and the basic fricative consonants from the features and then mapped into the corresponding static animation, called viseme (its counterpart in the visual domain [11]). In this frame-based processing mechanism, the latency is negligible as it almost equals the processing time of a single frame. However, a lack of consideration on neighborhood context information during the process may significantly limit the recognition accuracy and quality of the generated animation as the system can only recognize basic phonemes. In [9] and [10], a word-based processing mechanism is adopted to achieve much higher animation quality. By utilizing force alignment [21], phoneme transcription can be extracted from an audio chunk that contains multiple words with reasonably high accuracy. In [22], the neighboring phoneme pronunciation transition, named co-articulation, was considered. However, the latency of the word-level duration becomes unacceptable for real-time applications. In this article, we use a slide window of frames to obtain the corresponding phoneme at each timestamp. Our designed facial model addresses the co-articulation problem by considering the duration of the current phoneme's articulation and the corresponding associated phonemes before and after it.

III. System Components Design

In a typical real-time audio-driven speech animation system, the phoneme stream first is recognized from the audio input and then map to the corresponding parameter stream to derive the 3-D facial models.

Viseme is the visual mouth shape representation that has been widely used in speech recognition and animation, as it can be mapped directly from the phoneme. However, *vice versa* is not true since multiple phonemes may be mapped to the same viseme if they have similar mouth shapes during the pronunciation, such as /b/ and /p/). It is important to realize that so far, there is no common standard to regulate the viseme classes [23], for example, [24] used 20 different visemes, [25] used 26 different visemes, and [26] used 16 visemes, etc.

Fig. 1 gives a system overview of the proposed real-time audio speech animation systems that correspond to the phoneme stream {S, OW, LY} of vocabulary /slowly/. When the system receives an audio signal, it is transformed into the corresponding MFCCs features, which is the input of the RealPRNet. The RealPRNet predicts the phoneme stream and then maps it into the corresponding points (or blocks)

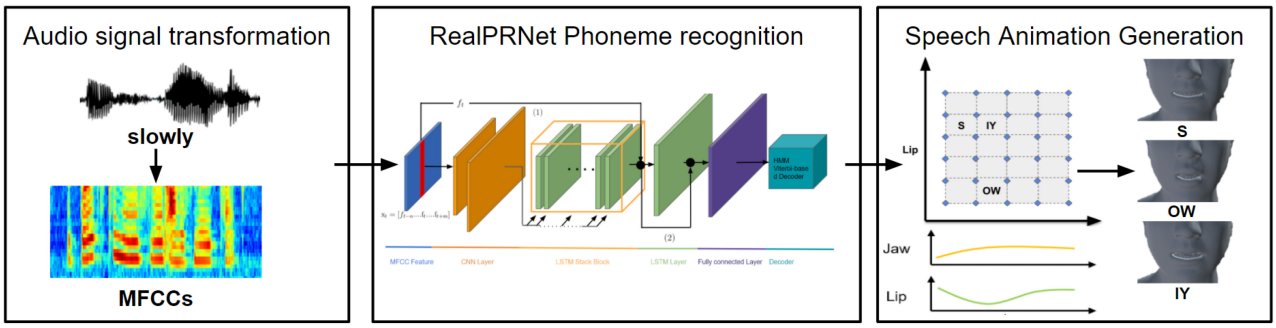

Fig. 1. System overview of the proposed real-time audio only speech animation system corresponds to the phoneme stream {S, OW, IY} of vocabulary /slowly/.

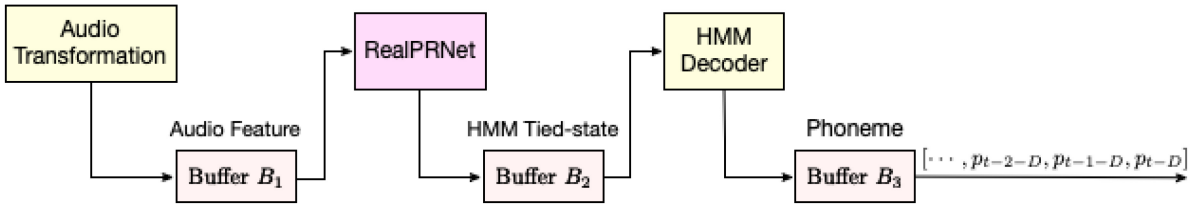

Fig. 2. Phoneme recognition system with buffers.

in the 2-D viseme field. The animation curve is shown in Fig. 1 connects the points on the 2-D viseme field and generates a parameter stream that can drive the 3-D facial model smoothly and seamlessly. To support real-time interactivities, the latency between receiving audio input and outputting animation is required to be controlled below certain thresholds (e.g., 200 ms [27], [28]) to ensure a believable and audience comfortable result. A buffer mechanism is adopted in the system to dynamically determine the size of the sliding window and ensure that the latency is within the required threshold. The four buffers are input feature buffer B_1, HMM tied-state output buffer B_2, output phoneme buffer B_3, and phoneme-selected buffer B_4.

A. Input Features Extraction Component

We employ the general audio process pipeline to extract the input audio features. When the raw audio signals are received in the system, they are transformed into frequency-domain signals, called MFCC, which is a data format that has been widely applied in automatic speech recognition (ASR). It is observed that the human voice is a combination of sound waves at different frequencies. The MFCCs can balance the variation of the sound change at different frequency levels. Generally, there are 100 frames/s in audio, and each audio frame is 25 ms with 15-ms overlap. For each audio frame, the first and the second derivative components of the MFCCs are aggregated to a vector that represents the single audio frame f. The input feature x_t at time t is transformed to the network feature f_t at time t surrounded by its forward and backward contextual vectors, denoted as $[f_{t-n}, \ldots, f_t, \ldots, f_{t+m}]$. The value of n represents the number of audio frames before time t, and the value of m represents the number of future audio frames. The integrated x_t is used to predict the phoneme label at time t. Thus, the value selection of m directly impacts the latency, which is $10m$ ms. The larger the value of m, the better potential recognition accuracy but longer latency. When $m = 0$, no future frames are buffered to introduce additional latency. However, the potential advantages of context information have been taken into consideration to improve the phoneme recognition performance.

B. Real-Time Phoneme Recognition Component Design

Real-time application systems need to be able to extract the audio features with high accuracy and low latency. Our proposed phoneme recognition scheme RealPRNet (described in Section III) can ensure a high-accuracy real-time recognition. The output of the RealPRNet is used as input to the HMM tied-state decoder H to calculate the output phoneme P. For the system to produce a smooth animation, the phoneme recognition system needs to be able to output the predicted phoneme with every A ms. Thus, a buffer mechanism is introduced in the system. An overview of the phoneme recognition system with buffers is shown in Fig. 2. All buffers in the figure are fixed size first-in–first-out (FIFO) buffers.

In the first step, the input raw audio signal is transformed into the network input feature f_t every A ms time interval (equal to the sampling interval), and f_t is stored in the input feature buffer B_1. The transformation of the raw audio signal to audio features causes the first delay $d_1 + e_1$, where d_1 is the median calculation time and e_1 is the corresponding fluctuation time. In our experiment, time consumption by this process is very stable and 100% less than A ms (10 ms in our experiment) as shown in Fig. 3 (blue).

All the audio frame features in B_1 are then used to construct x_t. The size of B_1 depends on two things: 1) the selected values m and n and 2) the RealPRNet time consumption $d_2 + e_2$ for each prediction. To guarantee a smooth output, the following inequality needs to be satisfied:

$$d_2 + e_2 \leq br \cdot A, \quad br = 1, 2, 3, \ldots \tag{1}$$

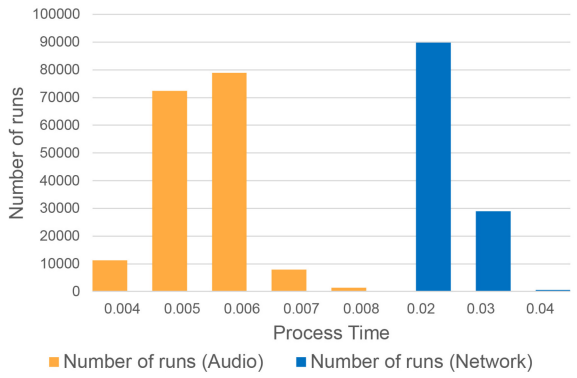

Fig. 3. Audio feature transformation time consumption distribution (blue), network prediction time consumption distribution (orange).

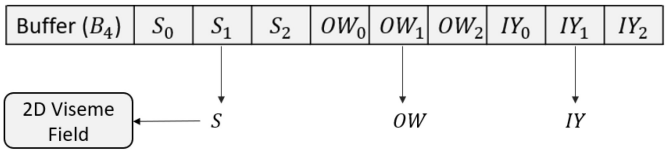

Fig. 4. Phoneme selection for 2-D viseme field animation curve generation.

br is the batch size used in the prediction progress. The neural network can parallelly predict br outputs in one run with a minimal increase in the computational overhead if br is a small value (i.e., $br = 10$). This is because when br is small, the input features' data size is relatively small compared to the RealPRNet's parameters. There is no sensible difference to today's computational power (e.g., in our experiment, one RTX 2080 Ti graphics card has been used) when processes single or br input features. The major time consumption is in data parsing and transmission. The RealPRNet takes the input features from B_1 every $br \cdot A$ ms and predicts br outputs $[h_t, h_{t-1}, \dots]$. These predicted outputs are stored in HMM tied-state buffer B_2. There are br sub-buffers in B_1 with size $m+n+1$ each and contents $f_{t-n-i}, \dots, f_{t-i}, \dots, f_{t+m-i}$. Because m forward audio frames are used to construct x, the system latency is increased to $m \cdot A + br \cdot A$ ms. In our experiment, the value of br is 4, which can ensure that (1) is satisfied in 99% of the cases. The network time consumption distribution is shown in Fig. 3 (orange).

The neural network in our phoneme recognition system does not directly predict phoneme labels but predicts HMM tied states instead which is because the combination of the neural network and HMM decoder can further improve the system performance. Such a kind of recognition system structure can be found in many related works [29], [30]. The predicted HMM tied states in B_2 are used as the input of the HMM triphone decoder. For each time interval $br \cdot A$, the decoder takes all the values in B_2, calculates the corresponding phonemes $[P_t, \dots, P_{t-br+1}]$, and stores it in B_3. The calculation time is $d_3 + e_3$ and the size of B_3 depends on the number of previous states used to calculate $[P_t, \dots, P_{t-br+1}]$. The HMM decoder improves the results by using the previously predicted tied state together with h_t to calculate the corresponding phoneme P_t. However, the decoder should only use the tied state as a reference rather than relying on it as in the speech recognition system because the phoneme-to-phoneme does not have a strong inner logic relation as word-to-word. Thus, the decoder is constructed with a bigram phoneme model, which focuses on the acoustic part (signal-to-phoneme) of the system rather than the language part (phoneme-to-phoneme) [12]. In our experiment, the calculation time consumed in decoding is stable and 100% less than $br \cdot A$. Thus, the overall latency from the raw input audio signal to the corresponding output phoneme is $(m + br) \cdot A + D + e_t$, where $D = d_1 + d_2 + d_3$ and $e_t = e_1 + e_2 + e_3$.

B_3 is the buffer that is used to control the final output of P of the phoneme recognition system. This output phoneme with a timestamp is stored into B_3. The system continuously takes the first phoneme from B_3 and uses it as the input to the animation system every A ms time interval. If a predicted phoneme with a timestamp has negative e_t, the system waits for A ms before the de-queued output from B_3. If a predicted phoneme P_{e_t} with a timestamp has positive e_t and the buffer contains no more phoneme, the system outputs the last phoneme and stores P_{e_t} as the last phoneme in the buffer. P_{e_t} can be used as an output if the next predicted phoneme also has a positive e_t. If not, the system drops P_{e_t} and then outputs the next predicted phoneme after a time interval of A ms. With this mechanism, the phoneme recognition system can have a stable output stream with a time interval A ms, and the overall latency from the raw input audio signal to the corresponding output phoneme stream is $(m + br) \cdot A + D$, where $D = d_1 + d_2 + d_3$. This stable output phoneme stream is stored in B_4. The following pseudocode represents this process. The `buffer` is B_3, and output `d_P` is the output phoneme, which is going to be stored in B_4.

The animation subsystem takes data in B_4 and uses it to produce the corresponding speech animation.

For the final speech animation generation, the output phoneme sequence of each time interval A ms from the previous component needs to be further processed. B_4 is used to select the appropriate next phoneme frame for the animation curve generation. As shown in Fig. 4, the same phoneme can occur in different frames. The size of B_4 is the average single phoneme pronunciation time in the audio frame. The output phoneme of the recognition system is stored in this buffer first. The phoneme pronunciation is dynamic progress, which means that the viseme phoneme at the beginning of the pronunciation is not the same as the corresponding phoneme viseme as shown in Fig. 5. Thus, an appropriate phoneme frame (for example, the phoneme frame can represent the rightmost one in Fig. 5 in a sequence of phoneme /o/ frames) should be selected from certain phoneme pronunciation frames to calculate the complete viseme's transformation time using the animation curve generation. If the minimum recognizable phoneme pronunciation time in the data set is $pr_{\min}$ audio frames and the length of the continuously predicted phoneme in the buff is less than $pr_{\min}$, then the corresponding phoneme will be replaced by the previous phoneme. The upcoming phoneme section rules can be described as follows.

1) The same phonemes are continuously appended to the buffer for at least $pr_{\min}$ units.

Fig. 5. Phoneme corresponding viseme at different frames during the pronunciation of phoneme /o/.

2) If the number of continuous phonemes is more than $pr_{\min}$ units and less than the size of B_4, the appropriate frame that represents the phoneme will be selected from that part of the buffer.
3) If the number of the continuous phonemes is more than the size of B_4, all phonemes in the buffer will be used to select the appropriate frame and no new frame will be selected until the next different phoneme is appended to the buffer.

C. Speech Animation Component Design

The speech animation component has been designed following the JALI viseme field and the state-of-the-art procedural lip-synchronization system [10]. The implementation in our system is slightly different from the original JALI viseme field. Most of the procedural systems, such as [10], [31], and [32], which use the keyframe viseme to produce the final animation have a similar problem, that is, the phoneme is mapped to one fixed static viseme without any variation. Through our observation of human speech behaviors, visemes corresponding to a phoneme may be slightly different in different situations. This is true even when the current phoneme has a sufficiently long pronunciation time to erase the co-articulation of the sound caused by the previous phoneme pronunciation. For example, the visemes that represent the phoneme /*s*/ pronounce differently in the *things* and *false* under the same speaking style. Thus, the 2-D viseme field has been divided into different blocks (20 blocks in our implementation), as shown in Fig. 6. Each phoneme corresponds to a block region rather than a static point in the 2-D field.

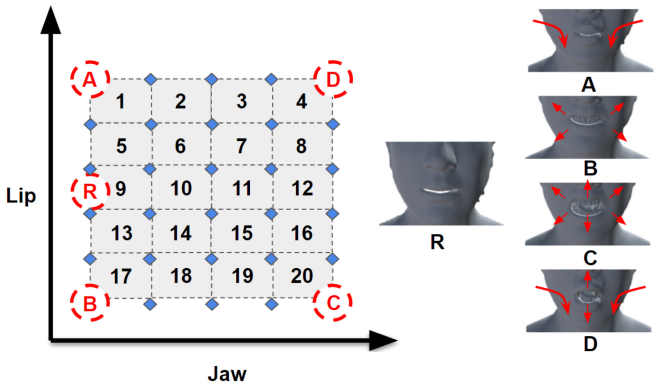

Fig. 6. 2-D viseme field with block index (left). The combination of the jaw and lip parameters in the extreme (A, B, C, D) and regular cases (R) visualized by the 3-D model viseme (right).

IV. Neural Network Design for Real-Time Phoneme Recognition

In recent years, neural network methods [33], [34] have dominated the phoneme recognition field and have dramatically improved recognition performance. We design a novel neural network architecture for our proposed real-time phoneme recognition task. The overall network architecture is shown in Fig. 7. The CNN layer takes x_t as the input and applies frequency modeling on the input features. The n-dimensional vector output of CNN is passed into an n-layer stack of LSTMs for parallel processing and temporal modeling. The output is combined with f_t to form an input to additional LSTM layers for temporal modeling and then goes through a fully connected layer. In the end, the HMM triphone decoder is adopted to predict the phoneme label.

A. CNN Layer

CNN has demonstrated outstanding performance in audio-related fields [34], [35]. The CNN layer is mainly used as a frequency modeling layer. An important contribution of the CNN layers is to reduce frequency variation. Based on the observation that voice of different people contains different ranges of frequencies even when they are speaking the same utterance, the traditional GMM/HMM-based speech recognition systems use techniques to reduce the frequency variation in the input feature, such as vocal tract length normalization (VTLN) [36] and feature space maximum-likelihood linear regression (fMLLR) [37]. It is reported recently [20] that CNN can provide the same feature improvement to the audio feature.

CNN layers also play an important role in the temporal–spatial domain [20]; here, spatial refers to frequency-domain audio feature pattern detection and learning. The input feature contains frequency information since each coefficient in Mel-frequency cepstral is generated by passing through different frequency filter banks. Each CNN filter can learn the different frequency patterns from the input features during the training. Our network architecture emphasizes these frequency patterns in the input features by separately connecting the CNN output features ($\mathrm{CNN}_{\mathrm{fout}}$) of different CNN filters to different LSTM in the LSB module. We use a 9×9 frequency (spatial)-temporal filter in the first CNN and a 3×3 filter in the second layer. The first layer has 256 output channels and the second layer has 16 output channels. We set no pooling in both CNN layers after observing that neither max nor average pooling helps to improve the result.

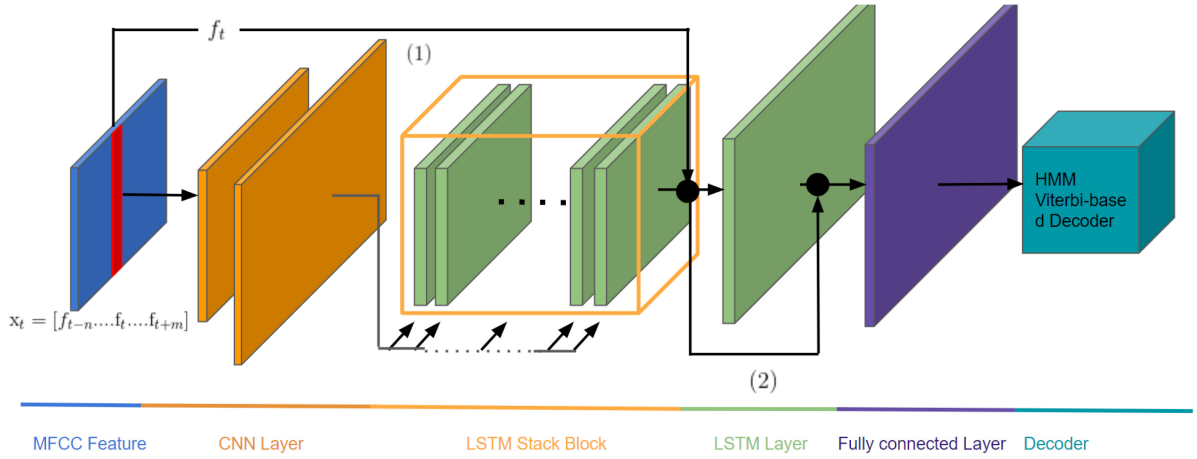

Fig. 7. Network architecture overview.

B. LSTM Stack Block

After frequency modeling is applied to CNN layers and the CNN filters have learned the acoustic features from the input feature, each CNN output channel produces the intermediate features as shown in Fig. 7. These features are applied to a parallel process of temporal modeling in LSB.

The $\mathsf{CNN}_{\mathsf{fout}}$'s from different CNN channels pass (2) to different LSTM modules, called the LSTM tube (LT), which is inside the LSB as shown in Fig. 7. The number of LTs inside the LSB depends on the number of the last CNN layer output channel, one LT for each CNN output channel. An LT is a relatively small and independent LSTM network. The LSTM has been proved to be advantageous on temporal modeling because it has memory cells to remember the information on previous features [19], [38]. Each output from different CNN channels can be viewed as a derivative feature of the original audio feature with the context within a sliding window. Thus, separate temporal modeling is applied to each different $\mathsf{CNN}_{\mathsf{fout}}$. The $\mathsf{CNN}_{\mathsf{fout}}$ is passed to LT0, where "0" represents the CNN filter with index 0, inside the LSB for independent temporal modeling (3) and the output feature is Lt_0. These separate output features are aggregated together as the input feature for the next LSTM layer (4)

$$\mathsf{CNN}_{\mathsf{outputs}} = \{C_{\mathsf{fout},1}, C_{\mathsf{fout},2}, \ldots, C_{\mathsf{fout},n}\} \quad (2)$$

$$\mathsf{Lt}_n = \mathsf{FCL}(\mathsf{LSTM}(\mathsf{CNN}_{\mathsf{fout},n})) \quad (3)$$

$$\mathsf{LSB}(\mathsf{CNN}_{\mathsf{outputs}}) = \{\mathsf{Lt}_1, \mathsf{Lt}_2, \ldots, \mathsf{Lt}_n\}. \quad (4)$$

In our network architecture, the LSB contains 16 LTs inside and each LT includes two LSTM layers and one fully connected layer. Each LSTM layer has 512 hidden units and a 0.3 dropout rate, and each fully connected layer has 128 output units.

C. LSTM Layer

In [12], it is demonstrated that the LSTM layer has its advantage in extracting temporal patterns in input feature space as we mentioned in the LSB. The gate units in LSTM are used to control the information flows inside the LSTM. The input gate decides what information can be added to the cell state. The forget gate decides what information needs to be removed from the cell state, and the output gate decides what information can be used in the output. Thus, the LSTM can selectively "remember" the temporal features. After the LSB performs the separate temporal modeling, we pass its output to the LSTM layers for the unified temporal modeling with the same logic. There are four LSTM layers, and each has 1024 hidden units, 512 output units, and a 0.2 dropout rate.

D. Fully Connected Layer

Following the state-of-the-art design of a typical deep learning network, we use the fully connected layer with softmax activation as the last layer so that a fully connected layer can provide the model with the ability to mix output signals from all neurons in the previous layer and the softmax can shape the output probabilities of each class so that the target class has a higher probability. The fully connected layer has 1024 hidden units. The output is an 1896-dimension vector, which represents the possibility of 1896 HMM-GMM tied states. The HMM triphone decoder takes this as an input to predict the final phoneme label.

E. Multiscale Features Addition

The idea of the multiscale feature addition was originally explored in computer vision and also used in ASR-related problems [35]. In a neural network, each layer focuses on different input features, varies from general to specific concepts. In ASR tasks, the lower layers (e.g., CNN layers in RealPRNet) focus more on speaker adaptation, and higher layers (e.g., LSTM layers in RealPRNet) focus more on discrimination [39]. Thus, the input features of different layers are complementarity, and the network's performance improvement has been observed by using these techniques [20]. In our implementation, we have explored two feature addition strategies, which are illustrated in Fig. 7 through line (1) and line (2), where the line (1) represents the original frame feature f_t in x_t combines with the LSB output and line (2) represents the LSB output combines with the LSTM output.

The first feature addition explores the complementary information in short-term feature f_t and long-term feature output feature from LSB (high-order representation of x_t). x_t is aggregated by using f_t and it's context features $[f_{t-n}, \ldots, f_{t-1}]$ and $[f_{t+1}, \ldots, f_m]$. However, the original LSTM does not

TABLE I
NETWORK PARAMETERS

	hidden units	output units	drop out rate	ksize
conv0	N/A	256	N/A	9
conv1	N/A	16	N/A	3
lsb0	1024	128	0.3	N/A
lstm0	1024	512	0.2	N/A
lstm1	1024	512	0.2	N/A
lstm2	1024	512	0.2	N/A
lstm3	1024	512	0.2	N/A
fcl0	1024	1896	N/A	N/A

consider their different values in prediction but takes all f's in x_t as consecutive features [40] and equally considers all the f's in x_t. This features addition emphasizes the importance of f_t in x_t when predicts p_t.

The second feature addition checks the LSB and the LSTM outputs, the separated and unified complementary temporal feature information. These features are the high-order feature representations of x_t with different pattern complementarity since network layers focus on different information in x_t with these patterns.

In the evaluation part, the performance improvement by multiscale feature addition has been explored, and the results show a positive effect on the network performance.

V. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the system and show that our proposed system can generate competitive speech animation results in real-time using only audio input, compared to a speech animation system using multimedia (video and audio) or offline method. The performance of the proposed system is evaluated in three areas: 1) the RealPRNet phoneme recognition accuracy; 2) the buffer occupancy dynamics to enable the real-time application; and 3) the subjective and objective speech animation quality.

A. Experiment Setup

Our experimental results are conducted using the TIMIT data set, which is widely used for phoneme recognition evaluation. It contains 6300 sentences, consisting of ten sentences spoken by 630 speakers each from eight major dialect regions of the United States. By following the standard TIMIT setup, we use the standard TIMIT training set, 3696 utterances from 462 speakers, to train our network and evaluate it on the TIMIT core test set, which consists of 192 utterances.

We use the TIMIT s5 recipe in Kaldi [41] to calculate phoneme duration in each utterance through force alignment technique and generate an HMM tied-state triphone model, the corresponding 1896 tied states and their properties (i.e., state probability, transfer probability, corresponding phoneme, etc.) We then use a triphone decoder with a bigram phone model in our system at the end of the neural network architecture, which takes the tied-states stream as input and outputs the predicted phoneme stream. The output ground truth y for the corresponding input feature x in the training data set is the index of the HMM tied states. Kaldi enabled audio to HMM tied-states force alignment. For the network training, ten epochs have been set as the minimum training epoch and to enable early stop (if the validate loss change in epochs is less than 0.001) during the training. The Adam optimizer was used in the first epoch and the momentum stochastic gradient descent (MSGD) optimizer for the rest epochs. The batch size is 256 for the first epoch and 128 for the rest. The learning rates are 0.01, 0.001, 0.0005, and 0.0001 for the first four epochs, and 0.0001 for the rest of them. The weight initialization has been used for the CNN layer's parameters, and the 0.3 dropout rate has been used for all the LSTM layers used (detail in Table I). The ReLU activation was applied to most of the layers except the final fully connected layer, which used softmax.

The performance of the network with a different m in the input feature was evaluated in this section. In our experiment, value n in x_t has been set to $m + 1$.

B. Error Metrics

1) Phoneme Error Rate: We first evaluate our proposed RealPRNet with a standard metric. During the evaluation, we first map the 60 phonemes to 39 and then calculate the "Levenshtein distance" between the recognized phoneme sequence and the ground-truth phoneme sequence. The Levenshtein distance is a method to measure the difference between the two sequences. It calculates the minimum number of single-character edits (including insertions, deletions, and substitutions) required to change one sequence into another. The number of edits required to change the recognized phoneme sequence into the ground-truth phoneme sequence is first calculated as the ratio between this minimum number of edits and the whole phoneme sequence length. This ratio is also known as the phoneme error ratio (PER). Under the real-time situation, the phoneme is predicted for each audio frame. Thus, the PER is calculated based on the audio frame-level phoneme sequence in our evaluation.

2) Block Distance Error: The block distance error (BDE) measures the average distance between the trajectory curve in the viseme field produced by the recognized phoneme sequence and the curve produced by the ground-truth phoneme sequence. The basic unit used here is the short edge of the 2-D viseme field block. For each audio frame, the corresponding points on the two curves and the absolute distance between these two points are calculated, also known as the Euclidean distance. Then, the average distance between these two curves at each time t was calculated

$$\mathsf{BDE} = \frac{1}{T}\sum_{0}^{T} P_{\mathsf{predict}} - P_{\mathsf{groundtruth}} \tag{5}$$

where P_{predict} is the predict phoneme position and $P_{\mathsf{groundtruth}}$ is the ground-truth position in the viseme filed. T is the total number of time intervals.

C. Phoneme Recognition Performance

In the experiment, RealPRNet was compared to two other phoneme recognition systems: 1) the 4-layer LSTM architecture presented in [12] and 2) the CLDNN [20].

Since most of the existing related work uses offline recognition schemes, we first evaluate the offline phoneme recognition

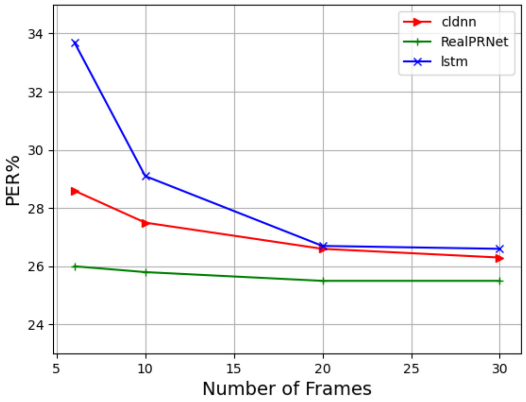

Fig. 8. PER per frames.

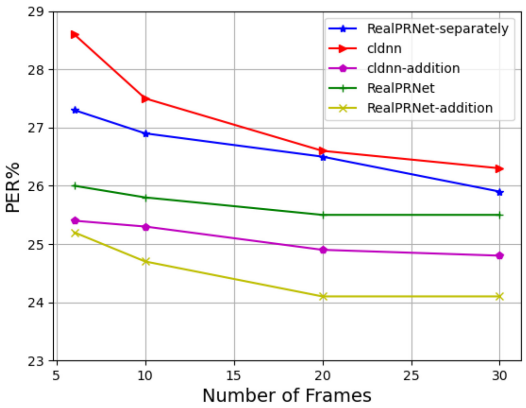

Fig. 9. Varies networks performance in PER.

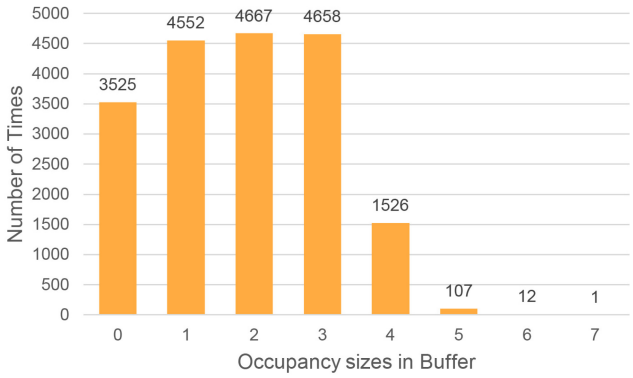

Fig. 10. Output phoneme buffer (B_3) occupancy.

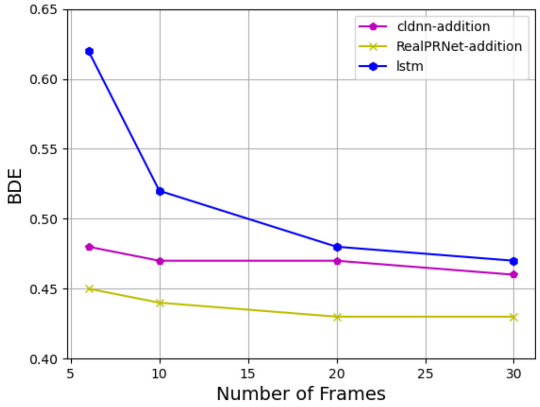

Fig. 11. BDE comparison.

capability of the baseline models. The best performance of the offline phoneme recognition of the above models in the evaluation using the TIMIT data set are 4-layer LSTM 18.63%, CLDNN 18.30%, and RealPRNet 17.20%. The RealPRNet outperformed other methods by 7.7% PER in the best case.

In Fig. 8, the real-time performance of these networks was compared. The x-axis in the figure represents the number of frames used in a single x_t, and the y-axis represents the phoneme error rate (PER). It is interesting to observe that LSTM has low performance when the temporal context information is insufficient (i.e., when x_t is aggregated by using less than 10 fs), but its performance improvement continuously with the increasing of the x_t value until a sweat spot (20 audio frame features in x_t) is achieved. In the figure, RealPRNet outperformed LSTM and for a minimum of 20% and 10%, respectively. The combination of frequency and temporal modeling enabled the RealPRNet a smooth performance across various selections of the number of frames for x_t.

To understand the advantage of RealPRNet further, we considered two additional variations: 1) RealPRNet without the LSTM layer and 2) CLDNN and RealPRNet without long short-term feature addition.

The RealPRNet applies two different types of temporal modeling to x_t, the separate temporal modeling performed by LSB and unified temporal modeling performed by LSTM. First, we explore the difference between these two temporal modelings that may affect the performance of the neural networks. The LSTM has been removed from the RealPRNet, and the new network is named "RealPRNet-separate." By comparing RealPRNet-separately and CLDNN, the result in Fig. 9 shows that the separate temporal modeling alone cannot outperform the unified temporal modeling. Thus, the performance of RealPRNet is benefited from when the two temporal modelings are used together. The PER of RealPRNet under different latency scenarios (different values m and n), shown in Fig. 9, illustrates that it outperforms both temporal modeling network architectures when used individually.

As illustrated in the previous section, the performance of the network can be further improved by multiscale feature addition techniques. We explore this technique with our RealPRNet and compare its performance with the strongest baseline CLDNN model with feature addition structure and also the previous networks without feature addition. As shown in Fig. 9, the performance of both RealPRNet and CLDNN with multiscale feature addition has been improved. The additional short-term feature gave complementary information to the intermediate input features, which forced the networks to focus on the current frame f_t. Thus, RealPRNet outperforms other networks in most of the latency scenarios. In particular, it can achieve an additional 4% relative improvement in the worst scenario in real-time PER evaluation.

D. Buffer Occupancy in Real-Time Application

The buffer mechanism is used to stabilize system output and ensure that the delay of the output is within the tolerance range. The occupancy status of B_3 of our experiment in runtime is shown in Fig. 10. In the ideal case, $[P_t, \ldots, P_{t-br+1}]$

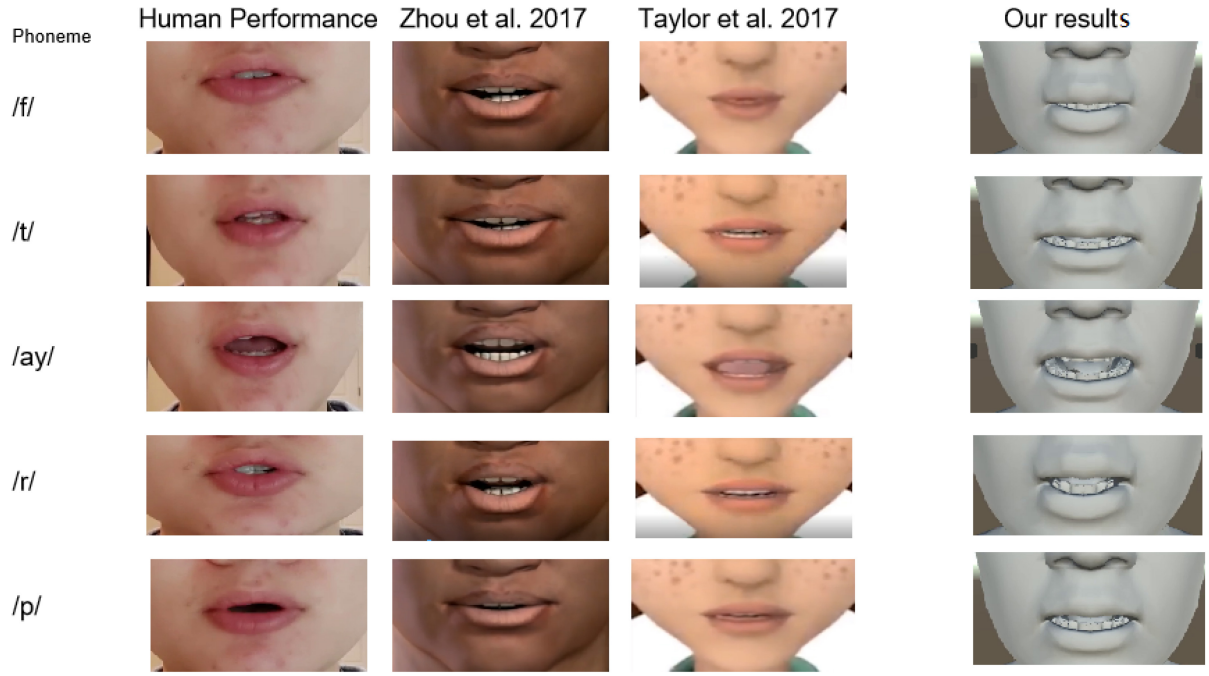

Fig. 12. Speech animation visual comparison with human performance and current state-of-the-art.

is queued to B_3 for every $br \times A$ ms, B_3 dequeues the first values for every A ms.

In our experiment, B_3 occupancy status is recorded after every dequeue, and the value of br is 4 in our experiment. Thus, B_3 occupancy sizes should be evenly distributed in $[0, 1, 2, 3]$. 90% of our test cases are in this range, which shows that our system can work in real-time situations. Most of the errors in B_3 occur in 0 and 4 (the number of occupied units) because the computer program is unable to use the exact A in each step during the runtime (e.g., the A fluctuated in a range from 10.3 to 11 ms). Other cases are caused by computational fluctuations of other parts of the phoneme recognition subsystems. This buffering mechanism ensures our proposed scheme is much more efficient to the extent that it can be implemented in real-time.

E. Animation Quality Assessment

In this section, we evaluate the quality of the output speech animation of our system.

First, we compare the trajectory curve in the viseme field produced by the recognized phoneme streams predicted by various reference networks and RealPRNet with the trajectory curve produced by the ground-truth model. The block short edge length is used as a distance unit to calculate the BDE between the recognized phoneme produced curve and the ground-truth curve at each time instance. The result is shown in Fig. 11, where the x-axis in the figure represents the number of frames used in input features and the y-axis represents the BDE in the basic unit. The RealPRNet also achieves the minimum error under different latency scenarios, which is 27% or less than other networks. In addition to the numerical evaluation, we also organized a group consists of 30 participants in total to watch our system-produced speech animation and ask them whether they feel the animation is believable or not (is the audio able to be synchronized with the 3-D model lip movement animation)? 29 out of 30 participants believe the audio has synchronized with the lip movement.

We also compare our result facial model's viseme with the current state-of-the-art's and real human performance. We take the viseme's screenshots at the peak of each phoneme pronunciation in an utterance to compare the results visually. As we can see in Fig. 12, all results are comparable.

VI. Conclusion and Further Discussion

In this article, we proposed an audio-driven believable speech animation production system with a new phoneme recognition neural network, called RealPRNet. The system can produce a competitive real-time speech animation with only raw audio input.

The RealPRNet has performed better than the strong baseline model in phoneme recognition, and the speech animation system is easy to be implemented in most of the existing virtual avatars. The real-human-like speech animation needs a lot of effort on the pretrain model with both video and audio data, even post edit from the artist. Our framework focuses on producing believable real-time speech animation with simple implementation, which also gives the user high freedom for online post effect editing.

For future work, we will consider the following issues: first, emotions may cause different mouth shape representations and mouth movements even with the same utterance. Knowledge of mouth movements under different emotions makes a more believable speech animation. Second, a believable speech animation involves the entire face movement, which can also be impacted through emotion. Third, the 2-D viseme field only contains two parameters to control jaw and lip movement. For high-quality speech animation of movies or AAA games, parameters (used to control the detailed mouth

shapes) should be discussed in more detail for each phoneme. Finally, we believe the tongue movement and visible parts of teeth are also very important in speech animation. With a similar mouth contour, different tongue and teeth statuses can also be produced.

REFERENCES

[1] K. Williams, R. Herman, and D. Bontempo, "Comparing audio and video data for rating communication," *Western J. Nurs. Res.*, vol. 35, no. 8, pp. 1060–1073, 2013.

[2] N. Sulaiman, A. M. Muhammad, N. N. D. F. Ganapathy, Z. Khairuddin, and S. Othman, "A comparison of students' performances using audio only and video media methods," *English Lang. Teach.*, vol. 10, no. 7, pp. 210–215, 2017.

[3] F. Tao and C. Busso, "End-to-end audiovisual speech recognition system with multitask learning," *IEEE Trans. Multimedia*, vol. 23, pp. 1–11, Feb. 2020.

[4] S. Taylor *et al.*, "A deep learning approach for generalized speech animation," *ACM Trans. Graph.*, vol. 36, no. 4, p. 93, 2017.

[5] M. Mehrabani, S. Bangalore, and B. Stern, "Personalized speech recognition for Internet of Things," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, 2015, pp. 369–374.

[6] M. Dawodi, J. A. Baktash, T. Wada, N. Alam, and M. Z. Joya, "Dari speech classification using deep convolutional neural network," in *Proc. IEEE Int. IOT Electron. Mechatronics Conf. (IEMTRONICS)*, 2020, pp. 1–4.

[7] M. La Mura and P. Lamberti, "Human-machine interaction personalization: A review on gender and emotion recognition through speech analysis," in *Proc. IEEE Int. Workshop Metrol. Ind. 4.0 IoT*, 2020, pp. 319–323.

[8] G. Llorach, A. Evans, J. Blat, G. Grimm, and V. Hohmann, "Web-based live speech-driven lip-sync," in *Proc. 8th Int. Conf. Games Virtual Worlds Serious Appl. (VS-GAMES)*, 2016, pp. 1–4.

[9] Y. Xu, A. W. Feng, S. Marsella, and A. Shapiro, "A practical and configurable lip sync method for games," in *Proc. Motion Games*, 2013, pp. 131–140.

[10] P. Edwards, C. Landreth, E. Fiume, and K. Singh, "JALI: An animator-centric viseme model for expressive lip synchronization," *ACM Trans. Graph.*, vol. 35, no. 4, p. 127, 2016.

[11] C. G. Fisher, "Confusions among visually perceived consonants," *J. Speech Hearing Res.*, vol. 11, no. 4, pp. 796–804, 1968.

[12] J. Michalek and J. Vaněk, "A survey of recent DNN architectures on the TIMIT phone recognition task," in *Proc. Int. Conf. Text Speech Dialogue*, 2018, pp. 436–444.

[13] S. Kapadia, V. Valtchev, and S. J. Young, "MMI training for continuous phoneme recognition on the timit database," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 2, 1993, pp. 491–494.

[14] I. Bromberg *et al.*, "Detection-based ASR in the automatic speech attribute transcription project," in *Proc. 8th Annu. Conf. Int. Speech Commun. Assoc.*, 2007, pp. 1829–1832.

[15] J. Morris and E. Fosler-Lussier, "Combining phonetic attributes using conditional random fields," in *Proc. 9th Int. Conf. Spoken Lang. Process.*, 2006, pp. 597–600.

[16] J. Park and H. Ko, "Real-time continuous phoneme recognition system using class-dependent tied-mixture HMM with HBT structure for speech-driven lip-sync," *IEEE Trans. Multimedia*, vol. 10, no. 7, pp. 1299–1306, Nov. 2008.

[17] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech, Signal Process*, vol. 37, no. 3, pp. 328–339, Mar. 1989.

[18] J. Kim, K. Hwang, and W. Sung, "X1000 real-time phoneme recognition VLSI using feed-forward deep neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2014, pp. 7510–7514.

[19] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 6645–6649.

[20] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2015, pp. 4580–4584.

[21] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using Kaldi," in *Proc. Interspeech*, 2017, pp. 498–502.

[22] P. L. Jackson, "The theoretical minimal unit for visual speech perception: Visemes and coarticulation," *Volta Rev.*, vol. 90, no. 5, pp. 99–115, 1988.

[23] H. L. Bear and R. Harvey, "Phoneme-to-viseme mappings: The good, the bad, and the ugly," *Speech Commun.*, vol. 95, pp. 40–67, Dec. 2017.

[24] Y. Zhou, Z. Xu, C. Landreth, E. Kalogerakis, S. Maji, and K. Singh, "Visemenet: Audio-driven animator-centric speech animation," *ACM Trans. Graph.*, vol. 37, no. 4, p. 161, 2018.

[25] C. Bregler, M. Covell, and M. Slaney, "Video rewrite: Driving visual speech with audio.," in *Proc. SIGGRAPH*, vol. 97, 1997, pp. 353–360.

[26] E. Bozkurt, C. E. Erdem, E. Erzin, T. Erdem, and M. Ozkan, "Comparison of phoneme and viseme based acoustic units for speech driven realistic lip animation," in *Proc. 3DTV Conf.*, 2007, pp. 1–4.

[27] P. Waddell, G. Jones, and A. Goldberg, *Audio/Video Synchronization Standards and Solutions a Status Report*, Adv. Television Syst. Committee, Washington, DC, USA, 1998.

[28] A. C. Younkin and P. J. Corriveau, "Determining the amount of audio-video synchronization errors perceptible to the average end-user," *IEEE Trans. Broadcast.*, vol. 54, no. 3, pp. 623–627, Sep. 2008.

[29] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, 2013, pp. 273–278.

[30] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," in *15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1964–1968.

[31] A. Pearce, B. Wyvill, G. Wyvill, and D. Hill, "Speech and expression: A computer solution to face animation," in *Proc. Graph. Interface*, vol. 86, Aug. 1986, pp. 136–140.

[32] S. A. King and R. E. Parent, "Creating speech-synchronized animation," *IEEE Trans. Vis. Comput. Graph.*, vol. 11, no. 3, pp. 341–352, May/Jun. 2005.

[33] K. Chen and Q. Huo, "Training deep bidirectional LSTM acoustic model for LVCSR by a context-sensitive-chunk BPTT approach," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 7, pp. 1185–1193, Jul. 2016.

[34] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014.

[35] S. Zhang, S. Zhang, T. Huang, and W. Gao, "Speech emotion recognition using deep convolutional neural network and discriminant temporal pyramid matching," *IEEE Trans. Multimedia*, vol. 20, no. 6, pp. 1576–1590, Jun. 2018.

[36] P. Zhan and A. Waibel, "Vocal tract length normalization for large vocabulary continuous speech recognition," Sch. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CS-97-148, 1997.

[37] M. J. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Comput. Speech Lang.*, vol. 12, no. 2, pp. 75–98, 1998.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] T. N. Sainath *et al.*, "Improvements to deep convolutional neural networks for LVCSR," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, 2013, pp. 315–320.

[40] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 338–342.

[41] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, 2011, pp. 1–4.

Zixiao Yu received the B.S. degree in electric and computer engineering from Michigan State University, East Lansing, MI, USA, in 2016, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

His research focuses on using deep-learning techniques to the automation of animation production, including real-time lip-synchronized animation generation, automatic camera planning with character actions' fidelity in 3-D scenes, and automatic camera planning with character emotion.

Haohong Wang received the B.S. degree in computer science and the M.Eng. degree in computer applications from Nanjing University, Nanjing, China, in 1994 and 1997, respectively, the M.S. degree in computer science from the University of New Mexico, Albuquerque, NM, USA, in 1998, and the Ph.D. degree in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 2004.

He is currently the General Manager with TCL Research America, TCL Corporation, San Jose, CA, USA, in charge of the overall corporate research activities in North America, including five research labs at multiple locations. Prior to that, he held various technical and management positions with AT&T, Dallas, TX, USA; Catapult Communications, Mountain View, CA, USA; Qualcomm, San Diego, CA, USA; Marvell, Wilmington, DE, USA; TCL-Thomson Electronics Corporation (TTE), Shenzhen, Guangdong, China; and Cisco, San Jose, CA, USA.

Dr. Wang was the Chair of IEEE Multimedia Communications Technical Committee from 2010 to 2012, and has been an Elected Member of the IEEE Visual Signal Processing and Communications Technical Committee since 2005, and IEEE Multimedia and Systems Applications Technical Committee since 2006. He has been the Editor-in-Chief of the *Journal of Communications* since 2008, the Technical Program Chair of IEEE GLOBECOM 2010 (Miami), and the General Chair of IEEE ICME 2011 (Barcelona) and IEEE ICCCN 2011 (Maui). He has served as the Editor-in-Chief of the IEEE MMTC E-Letter (2009) and as an Editor (or Guest Editor) of IEEE TRANSACTIONS ON MULTIMEDIA, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, *IEEE Communications Magazine*, and *ACM Multimedia Systems Journal*. He served as the General Chair of the 17th IEEE International Conference on Computer Communications and Networks in 2008 (U.S. Virgin Islands). He is a member of the Steering Committee of IEEE TRANSACTIONS ON MULTIMEDIA.

Jian Ren (Senior Member, IEEE) received the B.S. and M.S. degrees in mathematics from Shaanxi Normal University, Xi'an, China, in 1988 and 1991, respectively, and the Ph.D. degree in EE from Xidian University, Xi'an, in 1994.

He is a Professor with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA. His current research interests include cybersecurity, cloud computing security, distributed data sharing and storage, decentralized data management, blockchain-based e-voting and AI security, and Internet of Things.

Prof. Ren was a recipient of the U.S. National Science Foundation CAREER Award in 2009. He is serving as the TPC Chair of IEEE ICNC'17, the General Chair of ICNC'18, and an Executive Chair of ICNC since 2019. He currently serves as an Associate Editor for IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE INTERNET OF THINGS JOURNAL, and *ACM Transactions on Sensor Networks*, and a Deputy Editor-in-Chief for *IET Communications*.