Robust speech recognition based on deep learning for sports game review

Min Liu¹,†, AnYing-Hao¹, LiangFa-Li²

1. Sport Art College of Guangzhou Sports University, Guangzhou, 510500, Guangdong Province, China
2. Department of physical education, Guangdong University of foreign studies, Guangzhou, 510420, Guangdong Province, China

Abstract

To verify the feasibility of robust speech recognition based on deep learning in sports game review. In this paper, a robust speech recognition model is built based on the generative adversarial network GAN algorithm according to the deep learning model. And the loss function, optimization function and noise reduction front-end are introduced in the model to achieve the optimization of speech extraction features through denoising process to ensure that accurate speech review data can be derived even in the game scene under noisy environment. Finally, the experiments are conducted to verify the four directions of the model algorithm by comparing the speech features MFCC, FBANK and WAVE. The experimental results show that the speech recognition model trained by the GSDNet model algorithm can reach 89% accuracy, 56.24% reduction of auxiliary speech recognition word error rate, 92.61% accuracy of speech feature extraction, about 62.19% reduction of training sample data volume, and 94.75% improvement of speech recognition performance in the speech recognition task under noisy environment. It shows that the robust speech recognition based on deep learning can be applied to sports game reviews, and also can provide accurate voice review information from the noisy sports game scene, and also broaden the application area for deep learning models.

Keywords: Deep learning model; Generative adversarial network GAN algorithm; Robust speech recognition; Speech feature vector; Loss function

AMS 2020 codes: 68T05
1 Introduction

Since ancient times, speech has been efficient, easy to use, and more time-sensitive in conveying information [1]. Speech has unique advantages in conveying information and expressing emotions, and human civilization cannot be separated from the innovation of this information communication method. Speech recognition is a technology that converts audio into text by computer [2-4]. Since speech data are often of huge scale and impossible to interpret by human, converting massive audio files into text symbols can greatly save file storage space, improve efficient storage and transmission of information, and also make it easier to apply information in professional directions [5-6]. In the military field, professionals can further mine the intelligence data processed by this technology, compile key audio content and alert the key material in a timely manner, and help us grasp the development of the battlefield situation in a timely manner by deeply mining and effectively using the intercepted audio intelligence information [7]. In the civilian sector, this technology can be used for personal assistants such as Siri and Xiaoxiao classmates, and can also be used to assist the driver in directing the vehicle movement in autonomous driving technology, which is in the research stage [8-9].

Through the relentless exploration of researchers, milestones have been achieved while pushing the boundaries of technology. The literature [10] explored an interaction with gaze, gesture, and speech in a flexibly configurable augmented reality system. The literature [11] argues that noisy dataset expansion can improve noise robustness, but when noise reaches a certain level, noise reduction becomes more difficult. In addition, recognition of misclassified information in the back end [12] and fusion of noisy features with augmented features [13] have been proposed successively. The back-end is not only affected by the enhanced front-end, but also gets additional compensation from the original speech features. On this basis, the scheme of simultaneous back-propagation of enhanced recognition at the front and back-end has also been proposed [14]. The literature [15] proposes a multi-task learning architecture that learns multiple sources (i.e., speech and noise) and masks simultaneously [16]. The literature [17] proposed a multi-stream HMM architecture combining DNNs with traditional GMM-HMM models under noisy speech selection models. In [18], LAS is proposed and Sequence-to-Sequence is applied to speech recognition tasks for the first time. The literature [19] describes front-end based speech enhancement techniques and back-end based model adaptation techniques. Although research on the application of speech recognition is extensive, there are few studies on the application of robust speech recognition to review sports events.

This paper introduces the basic principle of the generative adversarial network GAN algorithm based on deep learning, introduces the activation function of the algorithm, improves the computing efficiency of the generative adversarial network GAN algorithm with the Adam optimization algorithm, and establishes a robust speech recognition model based on the generative adversarial network GAN algorithm under deep learning based on the shortcomings of the current robust speech recognition model. The robust speech recognition model based on the GAN algorithm of generative adversarial network with deep learning is established. The noise-reduced speech recognition is implemented with noise-reduced speech extraction front-end and speech recognition back-end design functions. Then, using simulated data as experiments, three expert-designed acoustic features: MFCC, FBANK, and WAVE are used to supervise the noise-reducing front-end extraction features, while the knowledge of the generative adversarial network GAN algorithm under deep learning is transferred to the neural network and generates a feature tensor that is more favorable for speech recognition by cross-training with the speech recognition back-end. Experimental data from four different directions are used to validate the application of robust speech recognition based on deep learning in sports games.
2 Deep learning model - generative adversarial network GAN algorithm

2.1 Fully connected neural networks

In machine learning, fully connected layers are often used for feature extraction and classification. A simple fully connected layer containing 4 neurons. A fully connected layer, where each node is connected to all nodes of the previous layer, is used to combine the features extracted from the previous side. Due to its fully connected nature, the fully connected layer generally has the most parameters as well.

In the fully connected layer, the neurons realize a linear function of for each neuron during the forward propagation computation. As shown in equation (1):

\[ y = f \left( \sum_{i=1}^{n} w_i x_i + b \right) = f(w^T x + b) \]  

(1)

Where, \( x_i \) is the input, \( x = (x_1, x_2, \cdots, x_n)^T \); \( w_i \) is the weight, \( w = (w_1, w_2, \cdots, w_n)^T \); \( b \) is the bias; \( f \) is the activation function; \( y \) is the output. The following four activation functions are often used in deep learning: Sigmoid function, Tanh function, ReLU function, and Leaky ReLU function:

\[ f(x) = \frac{1}{1 + e^{-\lambda x}}, \lambda > 0 \]  

(2)

The Sigmoid function maps real-valued variables to between 0 and 1, so it is widely used as the activation function of neural networks to complete the job of classification. However, the Sigmoid function saturates the gradient easily, which may lead to the instability of the algorithm. The Tanh function is a deformation of the Sigmoid function, and its mathematical formula is shown below:

\[ f(x) = \frac{2}{1 + e^{-\lambda x}} - 1 = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}}, \lambda > 0 \]  

(3)

In practical applications, the Tanh function has better convergence speed due to its 0-mean output, but there is still the problem of gradient disappearance. In order to avoid the problem that saturated functions tend to make the gradient disappear, the ReLU function has been widely used in recent years by virtue of its nonlinear and nonsaturated characteristics, and its mathematical formula is shown below:

\[ f(x) = \max \{0, x\} \]  

(4)

The ReLU function has the characteristics of simple computation and fast convergence, but for inputs less than 0, the parameters will not be updated during the gradient descent process, and the neurons will stop responding to the changes in the input, resulting in “death”:

\[ f(x) = \max \{\lambda x, x\} \]  

(5)
where \( \lambda > 0 \) (generally taken as 0.01), which preserves the gradient in the region of the definition domain less than 0, so that its information is not completely lost.

Deep learning models [20] use neurons to fit the data distribution, and using only one fully connected layer sometimes fails to solve the nonlinearity problem. To increase the complexity of the network, multiple fully connected layers are generally superimposed on the neural network to make the network perform better and the parameters, i.e., weights and biases, are updated using a back propagation algorithm.

2.2 Principle of Generating Adversarial Network GAN

Generative adversarial networks are [21] one of the deep generative models that have received more attention in the field of deep learning. The original GAN is based on the generative model with a dichotomous model, which is inspired by game theory, where the generative model and the dichotomous model will play a game during the training process, and the dichotomous model will guide the generative model to train in order to reach the Nash equilibrium. Here, the generative model is called the generator, denoted as \( G \), and the dichotomous model is called the discriminator, denoted as \( D \). The architecture of GAN is shown in Figure 1.

![Figure 1. Gan’s architecture](image_url)

The process of GAN training is the process of generators and discriminators confronting each other. The generator processes the random data so that the probability distribution of the data obeys a certain distribution and wants it to approximate the probability distribution of the real data. On the other hand, the discriminator learns the probability distributions of the real data and the generated data, and computes the probability that the input data comes from the real sample distribution. This result is then reverted to the generator to make it generate more realistic data. Eventually the discriminator does not distinguish the difference between the data generated by the generator and the real data, and then Nash equilibrium is reached.

From an information-theoretic perspective, cross-entropy is often used to measure the difference between two probability distributions. For two probability distributions \( p \) and \( q \) in a sample set, where \( p \) represents the true distribution and \( q \) represents the noise distribution, the formula to measure the relationship between these two distributions using cross-entropy is shown below:

\[
H(p, q) = \sum_x p(x) \log \left( \frac{1}{q(x)} \right)
\]

For continuous variables cross entropy formula is shown below:
In generative adversarial networks, for a discriminator, its loss function is expressed in terms of cross-entropy as follows:

\[ H(p,q) = \int x P(x) \log \left( \frac{1}{Q(x)} \right) dx = -E_p[\log Q(x)] \]  

(7)

Where: 
- \( E \) denotes the expectation; 
- \( x \) denotes the true sample data; 
- \( F_x(x) \) denotes the distribution of the true sample; 
- \( D(x) \) denotes the probability that \( D \) discriminates the true data as coming from the true sample; 
- \( z \) denotes the random data; 
- \( F_z(z) \) denotes the distribution that the random data samples obey; 
- \( G(z) \) denotes the generated data; and 
- \( D(G(z)) \) denotes the probability that \( D \) discriminates the generated data as coming from the true sample. The goal of \( D \) is to correctly determine the source of the data, so it wants \( D(x) \) to be close to 1 and \( D(G(z)) \) to be close to 0, while the goal of \( G \) is to make \( D(G(z)) \) closer to 1. Thus, similarly, the loss function of the generator is shown below:

\[ G_{\text{LOSS}} = E_{z\sim F_z(z)}[\log(1 - D(G(z)))] \]  

(8)

Thus, the optimization problem of GAN is transformed into a minimal-extreme game problem, and the objective function of the model is shown below:

\[ \min_{D} \max_{G} V(D,G) = E_{x\sim F_x(x)}[\log D(x)] + E_{z\sim F_z(z)}[\log(1 - D(G(z)))] \]  

(9)

The derivative of the above equation with respect to \( D(x) \) yields the extreme value:

\[ \frac{F_x(x)}{D(x)} - \frac{F_G(x)}{1 - D(x)} = 0, \quad D^*(x) = \frac{F_x(x)}{F_x(x) + F_G(x)} \]  

(10)

Where: \( F_G(x) \) indicates the distribution that the data samples generated by the generator obey. Theoretically, when the model is optimal, i.e., when the discriminator cannot distinguish whether the data comes from the true distribution, \( F_G(x) = F_x(x) \), \( D^*(x) = 0.5 \), the data generated by the generator is closest to the true data.

### 2.3 Algorithm design for generating adversarial networks

The Adam optimization algorithm is a first-order gradient-based algorithm for optimizing stochastic objective functions that dynamically adjusts the adaptive learning rate for each parameter based on the first-order moment estimates and second-order moment estimates of the gradient of the loss function [22] for each parameter.

Let the initial parameters of the model be \( \theta \), the objective function, i.e., the loss function, be \( f \), the exponential decay rates of the moment estimates be \( \beta_1 \) and \( \beta_2 \), and the general default be \( \beta_1 = 0.98, \beta_2 = 0.9998 \). Let the constant used for numerical stabilization be \( \varepsilon \), and the general default be \( 10^{-9} \). The Adam optimization algorithm is shown below:
Algorithm 1 Adam optimization algorithm [23]

Step 1. Initialize the first-order moment variables and the second-order moment variables 1, noting the number of iteration steps $t = 1$.

Step 2. Calculate the gradient of the model parameters $g_t = \nabla \theta f_t(\theta_{t-1})$.

Step 3. Update the biased first-order moment estimates $m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$ and biased second-order moment estimates $v_t = \beta_2 m_{t-1} + (1 - \beta_2)g_t^2$.

Step 4. Calculate the first-order moment deviation $\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$ and second-order moment deviation $\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$.

Step 5. Update the parameters: calculate $\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$.

If $\theta_t$ converges, output $\theta_t$; otherwise, set $t := t + 1$ and go to step 2.

The Adam optimization algorithm is applied to the model training of GAN, and the size of each data is 1, the number of each batch generation and input data is $n$, the parameter in the generator is $w_1$, the parameter in the discriminator is $w_2$, and the total number of training steps is epoch, then the algorithm of GAN is shown as follows:

Algorithm 2 GAN algorithm

Step 1. Randomly generate the batch noise data obeying Gaussian distribution, noting the number of iteration steps $t = 1$.

$$\{x_1^{(i)}\}_{i=1}^t, \{x_2^{(i)}\}_{i=1}^t, \ldots, \{x_n^{(i)}\}_{i=1}^t$$ \quad (12)

Step 2. Input the random noise data into the generator to generate the data.

$$\{G(x_1^{(i)})\}_{i=1}^t, \{G(x_2^{(i)})\}_{i=1}^t, \ldots, \{G(x_n^{(i)})\}_{i=1}^t$$ \quad (13)

Step 3. Obtain bulk real data from the dataset.

$$\{z_1^{(i)}\}_{i=1}^t, \{z_2^{(i)}\}_{i=1}^t, \ldots, \{z_n^{(i)}\}_{i=1}^t$$ \quad (14)

Step 4. Adam optimization: Using the loss functions of the generator and discriminator, the following equation is calculated.

$$w_1^t = \text{Adam}(w_1^{t-1}), w_2^t = \text{Adam}(w_2^{t-1})$$ \quad (15)
Step 5: If $t = epoch$, then the training is completed and the GAN model is output. Otherwise set $t = t + 1$ and go to step 2.

Step 6: Generate data: Randomly generate batch noise data obeying Gaussian distribution according to the number of generated data to be output.

$$\{y_1^{(i)}\}_{i=1}^l, \{y_2^{(i)}\}_{i=1}^l, \cdots, \{y_n^{(i)}\}_{i=1}^l \quad (16)$$

Generate and output data using the parameters of the trained generative model:

$$\{G(y_1^{(i)})\}_{i=1}^l, \{G(y_2^{(i)})\}_{i=1}^l, \cdots, \{G(y_n^{(i)})\}_{i=1}^l \quad (17)$$

3 Robust speech recognition model based on generative adversarial network algorithm

In the field of speech recognition, the application of neural networks to solve speech recognition problems has become a major trend due to their good generalization performance and their ability to be embedded in mobile devices. Its development has gone through stages from GMM-HMM to DNN-HMM and then to End-to-End. Meanwhile, when using the model in a practical application environment, multiple environmental noise interference still significantly affects the recognition effect. Therefore, in recent years, robust speech recognition technology has developed rapidly and formed two main research directions: front-end based speech enhancement technology and back-end based model adaptive technology.

3.1 Method flow and problem description

As shown in Figure 2, the entire methodological flow of the model and application [24] is:

![Diagram](image-url)
Step 1 Sports game speech dataset construction. Using private data and recorded data, combined with exogenous speech synthesis APIs to construct the sports game speech dataset.

Step 2 Data preprocessing. The time domain audio signal is extracted from its Mayer spectral coefficients [25] and FBANK features, as well as the time domain features of the speech signal, which are referred to as WAVE in this paper. The above three artificial features are used to represent the time and frequency domain features, which contain multidimensional information of the speech signal and portray the speech signal features from various aspects, which are convenient for the next training step.

Step 3 Network structure design. To make full use of the limited labeled data samples and improve the accuracy and robustness of model recognition. A feature extraction front-end network with self-supervised knowledge migration under the deep learning-based generative adversarial network algorithm GAN is designed. The purpose of extracting high-level features is achieved using one-dimensional convolution. And then the back-end of speech recognition is utilized to decode and predict characters by generative adversarial neural network with cluster search algorithm.

Step 4 Network model training. The overall cross-training method is used, with one batch of data input for each training round. The front-end is trained for feature extraction, and then the back-end is trained for speech recognition in the next batch, and the recognized characters are output and the loss values are calculated. After merging the loss values, the error back propagation algorithm is applied to calculate the gradient and update the model parameter values using the optimization algorithm, and iterate according to this method until the model converges.

Step 5 Sports game speech recognition test. After the model is trained, the accuracy test is completed using the test set, and finally the recognition results are used to complete the review of the sports game.

Wherein, throughout the steps, the problem of designing the network structure in step 3 can be described as: let \( \mathbf{x} \in \mathbb{R}^{n \times d} \) be the clean speech signal dataset, \( \mathbf{y} \in \mathbb{R}^{n \times l} \) be its corresponding labels used for contest critique, where \( n \) is the number of entries of the sample, and \( l \) is the length of each data entry. Given the training dataset \( \mathbf{D} = \{(x_i, y_i)\}_{i=1}^n \), there are \( n \) pairs of speech signals and utterances. The robust speech recognition problem can be described as a problem of finding a mapping: \( f_\theta : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times l} \) to map a speech signal to an utterance. Where \( f_\theta \) is determined by \( \theta \) parameters. The optimal parameter \( \theta \) is then found by solving the following optimization problem:

\[
\min_\theta \frac{1}{2} \sum_{i=1}^n \| f_\theta(x_i) - y_i \|_F^2.
\]  

(18)

However, in real application environments which are often full of noise and noisy mixing scenarios are complex, \( \text{Dist}(\mathbf{x}) \) is used to describe this mixing process. In this paper, a noise reduction front-end \( \mu_\alpha(d) \) is designed to extract features from noisy speech data, where \( \mu_\alpha \) is parameterized by \( \alpha \). Therefore, the whole noise reduction process can also be described as an optimization problem of finding an optimal parameter \( \alpha \). In summary, functions \( f_\theta \) and \( \mu_\alpha \) must be designed to implement the function of noise-reduced speech control, which requires denoising before recognition and finally finding the optimal parameters \( \theta, \alpha \) by solving the following optimization problem:
3.2 Model Design

In this section, functions $f_\theta$ and $\mu_\alpha$ are designed to implement the function of noise-reducing speech recognition. The overall architecture is shown in Figure 3, which consists of three parts: the first part is the noise addition module; the second part is the feature extraction front-end $\mu_\alpha$; and the third part is the speech recognition back-end $f_\theta$. Based on the modeling of the problem in the previous section, this section focuses on the design of the feature extraction front-end $\mu_\alpha$ function and the speech recognition back-end $f_\theta$ function.

To simulate a noisy environment, the clean data are first mixed with noise and then fed into a gated convolutional network of the generative adversarial network [26], which consists of gated one-dimensional convolutional modules embedded in an LSTM layer. Finally, the output feature tensor is fed to the knowledge migration modules: MFCC, FBANK and WAVE, which compute the loss function from the extracted feature language and the predicted features of the network and then perform back propagation to complete the self-supervised learning. The next batch of features is then sent to the speech recognition module for text label recognition, which are composed of 1D gated networks and GAN layers, respectively, and then back-propagated to complete the second training.
3.2.1 Noise addition

In order to simulate the effect of noise on game commentary under real sports game scene, this paper uses various noise addition intervention methods, after the noise is added, the speech signal can be described as the following equation:

\[
\text{Dist}(x) = p_i \cdot T(x)
\]

\[
T \in \{\text{Tem}(), \text{Rever}(), \text{Fre}(), \text{Add}()\}, x \in R_{pd}^{nl}
\]

where \( \text{Add}(\cdot) \) adds noise (e.g., alarm clock, knocking sound, phone ringing, and TV sound) to the original signal. \( \text{Fre}(\cdot) \) add interference by filtering the time signal through a stopband filter [27].
Rever(·) interference with the input signal through a 1300 shock response, which was originally applied from the image domain. \( p \) represents the possibility of some kind of intervention that eventually affects the original speech signal.

### 3.2.2 Feature extraction front-end

The feature extraction front-end \( \mu_\alpha \) is used to complete the feature extraction of the noise-added time domain signal spectrum. The front-end module can be divided into three parts, the first part is a gated convolutional network that introduces a residual structure to extract features and remove noise; the second part is an LSTM convolutional network; and the third part is three simple fully connected networks, called artificial knowledge migrators.

Gated convolutional networks under generative adversarial networks use a self-attentive mechanism. Experiments show that the gating mechanism allows nonlinear feature extraction and reduces gradient dispersion, while greatly increasing the computational speed of the convolutional network compared to the traditional convolutional network. Also, in order to increase the depth of the network while reducing information loss and gradient disappearance and gradient explosion, a residual structure [28] is introduced to construct a residual gated convolutional network, whose mathematical expression is:

\[
\begin{align*}
    h_t(X) &= (X * W + b) \odot \sigma(X * V + c) \\
    h_{t+1}(X) &= h_t(X) + h_o(X)
\end{align*}
\]

where \( X \) is a three-dimensional tensor that represents the input tensor in the input layer and is the output tensor of the previous layer. \( W \) denotes the convolution kernel.

The second part is an LSTM-based recurrent network whose basic structure has the mathematical expression:

\[
\begin{align*}
    i_t &= \mathcal{O}(U_i[h_{t-1}, x_t] + b_i) \\
    f_t &= \mathcal{O}(U_f[h_{t-1}, x_t] + b_f) \\
    o_t &= \mathcal{O}(U_o[h_{t-1}, x_t] + b_o)
\end{align*}
\]

where \( \mathcal{O} \) is the activation function, \( x_t \) denotes the vector consisting of the second dimension and the third dimension in the input matrix \( X \). \( U_i \) denotes the weight matrix of the input gate; \( U_f \) denotes the weight matrix of the forgetting gate; \( U_o \) denotes the weight matrix of the output gate, and \( b_i \), \( b_f \), and \( b_o \) denote the bias weights.

After that, the tensor is iterated from the LSTM output into the network called the artificial knowledge migration module. The artificial knowledge transfer module is three simple fully connected neural networks to fit three artificially extracted features in speech recognition domain: MFCC, FBANK and WAVE, where WAVE is a noise-free speech time domain signal. Since the FBANK features retain more of the original speech data compared to the MFCC features. Thus the three artificial features span the frequency and time domains and carry information with their own characteristics. Each network consists of a layer of fully connected network with 256 hidden units, and its training process is accomplished by solving the regression task, whose regression target is the relevant spectral
features extracted from clean speech, thus guiding the feature extraction front-end to automatically filter noise.

### 3.2.3 Speech recognition backend

Speech recognition backend \( f_{gb} \) implements the mapping of the extracted features to the recognized text. It consists of two parts, the first part is a residual gated convolutional module and the second part is a recurrent network based on LSTM. The \( f_{gb} \) network is composed of a series of deep residual gated convolutional units. However, the input tensor at this point is the features pre-trained by the feature extraction module, which is a two-dimensional tensor. After that, half of the channels are used to control the output of the other channels. Each layer of the gated network is connected by a cross-layer residual network to compensate for the loss of information and to reduce the gradient loss.

The structure of the whole decoding network consists of a two-layer LSTM network decoder. The output is transformed into probability values, which are described as predicted probabilities, and then its probabilities are transformed by the Softmax layer. Since the output is empty, the cluster search algorithm is used to decode Chinese characters in order to improve the decoding accuracy. The transformation process is shown in Eq:

\[
y_i = \frac{e^{b_i}}{\sum_{j=1}^{T} e^{b_j}}
\]

where the expression of the cluster search algorithm is:

\[
\arg\max_y \prod_{t=1}^{T_r} p(y^{<t\succ} | x, y^{<t\succ}, y^{<2\succ}, \cdots, y^{<T_r\succ})
\]

### 3.3 Robust sports game speech recognition model training based on deep learning

The feature extraction front-end of this model \( \mu_{\alpha} \) uses a self-supervised training method. In \( \mu_{\alpha} \), after extracting high-level features from the noise-added pure original speech signal, the high-level features are outputted by three small simple fully connected networks respectively. And then the output of the fully-connected network is computed with the artificial features with cross-entropy loss, so that the three simple fully-connected layers output can fit the artificial features and make the back-propagation affect the high-level features. In this way, with the same number of training samples, the artificial features of the extracted training samples are self-supervised to train the advanced features extracted by the neural network, and realize the migration of artificial knowledge to the advanced features extracted by the feature extraction front-end \( \mu_{\alpha} \). The advanced features can fuse the advantages of the three artificial features to generate a feature tensor more favorable to speech recognition, which improves the utilization of training samples and increases the robust speech recognition accuracy.

Speech recognition backend \( f_{gb} \) uses supervised learning, which is divided into two stages: forward propagation and backward propagation. The forward propagation process uses the original information to perform matrix operations along the designed network structure, and the network calculates a target value and the corresponding loss function value after each forward propagation, which is a measure of the gap between the network output target value and the sample labels. And
then, according to the error value, the back propagation algorithm is applied to calculate the gradient value of each parameter, and the Adam algorithm is used to update the optimization function to update the model parameters. At this point, a batch of training is completed, and the above process is iterated in turn until the completion of the specified number of iterations, the network model training is completed, and the data training on the training set is called an Epoch once. The training structure is shown in Figure 4.

During the cross-training between \( \mu_a \) and \( f_\theta \), the overall loss function of the model is:

\[
L = L_{f_\theta} + L_{\mu_a}\tag{26}
\]

Where, \( L \) denotes the loss function of the feature extraction front-end, which is composed of \( L_{\mu_a} = L_{mfcc} + L_{fbank} + L_{wave} \), where, \( L_H = H(x_i) \) is one of the loss functions in the process of fitting artificial features composed of three artificial features: MFCC, FBANK, and WAVE, where, \( L_{mfcc} \) is the loss function of fitting MFCC features, \( L_{fbank} \) is the loss function of fitting FBANK features, and \( L_{wave} \) is the loss function of fitting WAVE features. The expression of its loss function is:

\[
L_H = || \mu_a(x_i) - H(x_i) ||_F^2 \tag{27}
\]

\( L_{f_\theta} \) is the loss function of the back-end of speech recognition, and its expression is:

\[
L_{f_\theta} = || f_\theta(D) - y_i ||_F^2 \tag{28}
\]

The gradient update equation is:

\[
\begin{align*}
G_{f_\theta} &= \nabla_{f_\theta} L \\
G_{\mu_a} &= \nabla_{\mu_a} L
\end{align*} \tag{29}
\]
4 Experimental setup and analysis

4.1 Experimental data set

The method of exogenous API speech synthesis and recording method were used in this experiment. The following open-source speech datasets were used as test sets: THCHS-30, Hill Shell dataset and ST-CMDS. The loss functions were CTC loss function and Adam optimization function [29-29]. The word error rate algorithm is referred to the literature [30-31].

4.1.1 Feature extraction front-end setup

As shown in Table 1, the feature extraction front-end $\mu_\alpha$ of the feature extraction network consists of 8 layers of residual gated convolutional blocks, where each residual is passed through the convolutional block to form a gate, followed by a Weight Norm layer connected for parameter regularization before feeding into the convolutional block, and then a Dropout to improve the generalization of the network. The residual gated convolutional network structure can be defined as (number of input channels, number of output channels, 1D convolutional kernel size, convolutional step size, padding size), where the padding size is an optional parameter. The input speech signal size parameters are defined as (number of samples in batch, number of channels, dimensionality of the feature vector).

<table>
<thead>
<tr>
<th>Module structure</th>
<th>Dimension parameters</th>
<th>The amount of parameters/MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the tensor</td>
<td>$(30,1,31000)$</td>
<td>30</td>
</tr>
<tr>
<td>Gated Block 1</td>
<td>$(1,65,1,1)$</td>
<td>65</td>
</tr>
<tr>
<td>Gated Block 2</td>
<td>$(64,64,25,15)$</td>
<td>4096</td>
</tr>
<tr>
<td>Gated Block 3</td>
<td>$(64,128,12,1)$</td>
<td>8192</td>
</tr>
<tr>
<td>Gated Block 4</td>
<td>$(128,128,11,2)$</td>
<td>16384</td>
</tr>
<tr>
<td>Gated Block 5</td>
<td>$(128,256,12,1)$</td>
<td>32768</td>
</tr>
<tr>
<td>Gated Block 6</td>
<td>$(256,256,11,2)$</td>
<td>65536</td>
</tr>
<tr>
<td>Gated Block 7</td>
<td>$(256,512,11,1)$</td>
<td>131072</td>
</tr>
<tr>
<td>Gated Block 8</td>
<td>$(512,512,11,2)$</td>
<td>262144</td>
</tr>
<tr>
<td>LSTM</td>
<td>512</td>
<td>-</td>
</tr>
<tr>
<td>MFCC</td>
<td>1220</td>
<td>-</td>
</tr>
<tr>
<td>FBank</td>
<td>1220</td>
<td>-</td>
</tr>
<tr>
<td>WAVE</td>
<td>1220</td>
<td>-</td>
</tr>
</tbody>
</table>

4.1.2 Speech recognition back-end settings

As shown in Table 2, the speech recognition backend $f_\theta$ consists of a residual gated convolutional network followed by an LSTM with a cluster searcher. The input tensor is the self-supervised trained high-level features extracted by the feature extraction front-end $\mu_\alpha$, whose parameters are defined as (number of Batches, word embedding tensor, number of output words). The LSTM parameters are defined as (the dimension of the input tensor, the hidden layer dimension, and the number of layers of the LSTM). The hidden layer dimension is a dictionary dimension, which is equal to the dimension of the label dictionary. The final output tensor is the probability after passing through the Softmax function.
layer, and a cluster search algorithm is used to perform a clustering search to predict three potential words in parallel.

<table>
<thead>
<tr>
<th>Identify the backend</th>
<th>Dimension parameters</th>
<th>The amount of parameters/MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>(64,160,600)</td>
<td>6.01</td>
</tr>
<tr>
<td>Gated Block</td>
<td>(160,500,50,2,98)</td>
<td>3.9</td>
</tr>
<tr>
<td>7*Gated Block</td>
<td>(250,500,8,2)</td>
<td>6.07</td>
</tr>
<tr>
<td>Gated Block</td>
<td>(250,2000,30,2)</td>
<td>15</td>
</tr>
<tr>
<td>Gated Block</td>
<td>(1000,2000,1,1)</td>
<td>2</td>
</tr>
<tr>
<td>Convld</td>
<td>(1000,Output Units,2,1)</td>
<td>-</td>
</tr>
<tr>
<td>Intermediate Tensor</td>
<td>(64,1000,Output Units)</td>
<td>-</td>
</tr>
<tr>
<td>LSTM</td>
<td>(1024,Dictionary Dim,1)</td>
<td>-</td>
</tr>
<tr>
<td>Softmax</td>
<td>(Output Units, Dictionary Dim)</td>
<td>-</td>
</tr>
<tr>
<td>Beam Search</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

4.1.3 Comparative experimental setup

Firstly the performance is compared by models with and without artificial knowledge migration module. Secondly the speech recognition accuracy is compared by comparing the features extracted from the self-supervised training with the traditional manual features MFCC. Thirdly by setting up comparison experiments with and without self-supervised knowledge migration to investigate the difference in data cost used to achieve the same accuracy. Finally, the experimental comparison demonstrates the contribution of the training process using pre-training method and cross-training method to improve the speech recognition performance.

4.2 Results and Analysis

Under the experimental setup in the previous section, the experimental results are analyzed and discussed in four aspects, namely, the structure of generative adversarial networks under deep learning, the impact of generative adversarial network knowledge migration on feature extraction, the impact of generative adversarial network knowledge migration on training cost requirements, and the performance comparison with other novel algorithmic architectures.

4.2.1 Effect of Generative Adversarial Network Structure on Accuracy under Deep Learning

As shown in Table 3, the performance of the models was improved to different degrees by adding different network modules for the pure and noise-added signals of the three datasets THCH-30, AISHELL-1, and ST-CMDS. Among them, by adding the new deep learning adversarial network knowledge migration module, compared with the structure without adding artificial features, it illustrates that the deep learning adversarial network knowledge migration plays a better role in assisting speech recognition, and the speech recognition word error rate is reduced by 56.24%. Where gated cnn stands for gated convolutional network, skip connection is the residual structure, and new workers are the three fully connected networks added and self-supervised training. The performance parameters in the table are word error rates.
Table 3. Structural module comparison

<table>
<thead>
<tr>
<th>Structural changes</th>
<th>THCH-30 Clean</th>
<th>THCH-30 Noise</th>
<th>AISHELL-1 Clean</th>
<th>AISHELL-1 Noise</th>
<th>ST-CMDS Clean</th>
<th>ST-CMDS Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base structure</td>
<td>0.314</td>
<td>0.362</td>
<td>0.314</td>
<td>0.390</td>
<td>0.438</td>
<td>0.561</td>
</tr>
<tr>
<td>+Gated cnn</td>
<td>0.200</td>
<td>0.229</td>
<td>0.238</td>
<td>0.257</td>
<td>0.431</td>
<td>0.447</td>
</tr>
<tr>
<td>+50 hours</td>
<td>0.133</td>
<td>0.162</td>
<td>0.134</td>
<td>0.172</td>
<td>0.155</td>
<td>0.257</td>
</tr>
<tr>
<td>+Skip connection</td>
<td>0.181</td>
<td>0.219</td>
<td>0.219</td>
<td>0.238</td>
<td>0.419</td>
<td>0.390</td>
</tr>
<tr>
<td>+New workers</td>
<td>0.162</td>
<td>0.143</td>
<td>0.124</td>
<td>0.143</td>
<td>0.153</td>
<td>0.200</td>
</tr>
</tbody>
</table>

4.2.2 Effect of Generative Adversarial Network Structure on Feature Extraction under Deep Learning

As shown in Table 4, the use of different extracted features under pure and noisy signals from three datasets THCH-30, AISHELL-1, and ST-CMDS played different degrees of influence on the performance of the model. The number of covariates in Table 4 is the word error rate by artificial features MFCC, FBANK, WAVE compared with the feature extractor trained by structural knowledge migration of generative adversarial networks under deep learning, the latter achieved better robustness and accuracy, and 92.61% accuracy. The above comparison experiments demonstrate that the introduction of feature extraction networks under deep learning can utilize the features of multiple artificial knowledge to generate advanced features that are more beneficial for speech recognition.

Table 4. Feature extraction performance comparison

<table>
<thead>
<tr>
<th>Experimental projects</th>
<th>THCH-30 Clean</th>
<th>THCH-30 Noise</th>
<th>AISHELL-1 Clean</th>
<th>AISHELL-1 Noise</th>
<th>ST-CMDS Clean</th>
<th>ST-CMDS Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCC</td>
<td>0.296</td>
<td>0.325</td>
<td>0.211</td>
<td>0.249</td>
<td>0.221</td>
<td>0.458</td>
</tr>
<tr>
<td>FBANK</td>
<td>0.315</td>
<td>0.410</td>
<td>0.220</td>
<td>0.315</td>
<td>0.315</td>
<td>0.505</td>
</tr>
<tr>
<td>WAVE</td>
<td>0.334</td>
<td>0.439</td>
<td>0.230</td>
<td>0.372</td>
<td>0.382</td>
<td>0.581</td>
</tr>
<tr>
<td>GSDNet+(Supervised)</td>
<td>0.144</td>
<td>0.173</td>
<td>0.154</td>
<td>0.178</td>
<td>0.174</td>
<td>0.277</td>
</tr>
<tr>
<td>GSDNet+(Finetuned)</td>
<td>0.135</td>
<td>0.154</td>
<td>0.144</td>
<td>0.169</td>
<td>0.165</td>
<td>0.220</td>
</tr>
<tr>
<td>GSDNet+(Frozen)</td>
<td>0.147</td>
<td>0.182</td>
<td>0.149</td>
<td>0.182</td>
<td>0.173</td>
<td>0.287</td>
</tr>
</tbody>
</table>

4.2.3 Impact of Generative Adversarial Network Structure under Deep Learning on Training Cost Requirements

As shown in Figure 5, the GSDNet model is compared with the model featuring MFCC and the model featuring FBANK in terms of data volume and performance improvement. The amount of data used by the model to achieve different accuracies during training with and without knowledge migration of the generative adversarial network under deep learning is plotted. It can be concluded that the amount of training data is reduced by about 62.19% with the introduction of knowledge migration for generative adversarial networks with deep learning for the same accuracy. It shows that generative adversarial networks with deep learning can reduce the use of data cost and make full use of the value of sample data.
4.2.4 Performance comparison with other novel algorithmic architectures

As shown in Figure 6, the experimental comparison of this paper’s speech recognition algorithm based on generative adversarial network under deep learning with LAS and CTC algorithms in speech recognition shows that GSDNet has the highest accuracy and robustness of 94.75% for the same group under both clean and noise-added processing in three datasets THCH-30, AISHELL-1 and ST-CMDS. It shows that the algorithm in this paper can be good at identifying key speech in noisy sports games in robust speech recognition based on deep learning, and also provides critical and accurate information for reviewing sports games.
5 Conclusion

In order to verify the application of robust speech recognition technology of deep learning in sports game review. In this paper, a robust speech recognition model based on the generative adversarial network GAN algorithm with deep learning is established. For the problems of sample noise interference and limited data samples in the process of sports commentary speech recognition, a robust speech recognition algorithm based on deep learning self-supervised knowledge migration is proposed, which supervises the noise removal front-end extraction features by three expert-designed acoustic features: MFCC, FBANK, WAVE, and also cross-trains with the speech recognition back-end, and the deep learning-based generative Confrontation Network GAN algorithm knowledge is migrated to the neural network and generates a feature tensor that is more beneficial for speech recognition. After the experiments, it is shown that after the training of the integrated dataset, the speech recognition model trained by GSDNet can reach 89% accuracy in the speech recognition task in noisy environment, the auxiliary speech recognition word error rate is reduced by 56.24%, the speech feature extraction accuracy reaches 92.61%, the training sample data volume is reduced by about 62.19%, and the speech recognition performance is improved to 94.75%. It shows that robust speech recognition based on deep learning can well identify sports game reviews in noisy environment and also extract critical review information from the noisy sports game scene to provide accurate data for sports game broadcast.

References


