Sheferaw *et al.*
*EURASIP Journal on Audio, Speech, and Music Processing* (2024) 2024:6
https://doi.org/10.1186/s13636-023-00325-3

EURASIP Journal on Audio, Speech, and Music Processing

METHODOLOGY

Open Access

# Gated recurrent unit predictor model-based adaptive differential pulse code modulation speech decoder

Gebremichael Kibret Sheferaw[1*], Waweru Mwangi[1], Michael Kimwele[1] and Adane Mamuye[2]

**Abstract**

Speech coding is a method to reduce the amount of data needs to represent speech signals by exploiting the statistical properties of the speech signal. Recently, in the speech coding process, a neural network prediction model has gained attention as the reconstruction process of a nonlinear and nonstationary speech signal. This study proposes a novel approach to improve speech coding performance by using a gated recurrent unit (GRU)-based adaptive differential pulse code modulation (ADPCM) system. This GRU predictor model is trained using a data set of speech samples from the DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus actual sample and the ADPCM fixed-predictor output speech sample. Our contribution lies in the development of an algorithm for training the GRU predictive model that can improve its performance in speech coding prediction and a new offline trained predictive model for speech decoder. The results indicate that the proposed system significantly improves the accuracy of speech prediction, demonstrating its potential for speech prediction applications. Overall, this work presents a unique application of the GRU predictive model with ADPCM decoding in speech signal compression, providing a promising approach for future research in this field.

**Keywords** Speech coding, Gated recurrent unit, Nonlinear prediction, Waveform coding, Audio coding, Adaptive differential pulse code modulation, Speech compression

## 1 Introduction

Speech coding is the process of converting a speech signal into a more compressed form of digital data [1]. Then, it can be transmitted with fewer bits or saved and reconstructed into the original speech signal [2, 3].

Speech coding studies have a number of specific goals such as low compression, quality, lower delay, stability, compatibility, complexity, and scalability. The quality of the decoded speech signals should be as close to the original speech signals as possible.

Speech coding technologies can generally be divided into three main categories: waveform coding, vocoder coding [4], and hybrid coding [3, 5]. Waveform coding [6] is a technique used to represent and compress speech signals. This involves digitizing the analog speech waveform and encoding them into digital format to store or transmit them. Some commonly used waveform coding techniques include pulse code modulation (PCM) and adaptive differential pulse code modulation (ADPCM). ADPCM is a commonly used audio coding technique that achieves compression by predicting and quantizing the difference between consecutive samples. There exist several types of ADPCM, including IMA-ADPCM, Microsoft ADPCM, DVI4-ADPCM, Intel/DVI ADPCM, Yamaha ADPCM, Dialogic ADPCM, and others.

*Correspondence:
Gebremichael Kibret Sheferaw
gebrsh@gmail.com
[1] School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya
[2] School of Information Technology and Engineering, Addis Ababa University Institute of Technology, Addis Ababa, Ethiopia

Additionally, there are several ITU-T standards such as G.726, G.727, G.723, G.723.1, among others.

In our study, we have used IMA ADPCM as the foundational approach to improve the quality of audio coding. This improvement is achieved by integrating our proposed GRU predictive model into the IMA ADPCM framework. The selection of IMA ADPCM among the various ADPCM types and ITU-T standards due to its simplicity of use [7], low computational complexity [8], and suitability for real-time communication systems. And also IMA-ADPCM is widely used and compatible with our proposed integration strategy and incorporates adaptive quantization, enabling dynamic adjustment of the quantization step size to optimize coding performance for a broad range of audio signals. Thus, IMA-ADPCM aligns well with our objective of embedding or enhancing decoding predictive models for superior audio coding quality.

The IMA-ADPCM speech coding algorithm includes significant encoding and decoding processes. The algorithm begins with the encoding process and ends with the decoding function. In the IMA-ADPCM encoding process, the algorithm takes a 16-bit PCM speech sample and compresses it to a 4-bit value [9, 10] using adaptive quantizer and fixed predictor. The difference between the current and previous sample is calculated and the data is quantized to a new sample value using a variable step size. The resulting ADPCM code is then encoded using the quantizer step size. The predicted speech sample and quantizer step size [11] from the previous iteration are restored. The encoder generates a 4-bit ADPCM code based on the difference sample and step size.

In the ADPCM decoding process, the 4-bit ADPCM code is received and used to generate a predicted speech sample [11]. The step size index is used to determine the quantizer step size from a table. The 4-bit code is inverse-quantized, and the new speech sample value is computed by adding it to the previous predicted speech sample value. The step size index is updated on the basis of the modifications. The decoding system generates a new 16-bit sample by adding the difference to the previous prediction.

During the encoding and decoding procedures, the ADPCM algorithm adjusts the quantizer step size based on the previous ADPCM value. This adjustment is calculated using a step size calculation equation and implemented using lookup tables. The quantizer adaptation process ensures that the appropriate step size is used for each sample, taking into account the magnitude of the ADPCM code.

Speech coding prediction is a means of using some or all past speech samples to predict the present sample [12]. It is widely used in speech coding approaches such as ADPCM, LPC, and CELP. The speech coding prediction approaches are classified as linear or nonlinear prediction models.

The linear predictive technique [13] is well known and well understood, making simulation and implementation easy. But it is likely to be less powerful than a nonlinear prediction model. However, nonlinear speech prediction has recently attracted a lot of attention since the generation of the speech signal is a nonlinear and nonstationary process [12]. So far, the most commonly used nonlinear methods for nonlinear and nonstationary speech prediction have been classified into two categories: neural networks and polynomial filters.

### 1.1 Neural network-based speech coding

Neural network-based speech coding methods has seen significant advancements in recent years, with applications of deep neural networks (DNNs) and other neural architectures. These methods offer the advantage of learning relevant patterns automatically from large datasets, leading to improvements in speech quality, compression efficiency, and computational complexity.

WaveNet [14, 15], a deep generative model based on autoregressive CNNs, achieves high speech quality but suffers from high computational complexity and long inference times, limiting its real-time applications. WaveRNN [16], combining RNNs with ResNet, maintains high-quality speech synthesis with lower complexity than WaveNet. LPCNet [17], a combination of linear predictive coding (LPC) and WaveRNN, excels at low bit-rate speech coding, making it suitable for limited bandwidth applications. WaveGlow [18], a flow-based generative model, offers faster synthesis times than autoregressive models but may sacrifice fine-grained speech details. Parallel WaveGAN [19], employing inverse autoregressive flow (IAF), provides real-time generation with improved computational efficiency and good speech quality.

Parametric-based NN speech coding methods represent speech using parametric features rather than directly encoding waveforms. DNN-HMM [20] hybrid systems combine DNNs with HMMs for improved modeling capabilities and long-term dependencies in speech. Variational autoencoders (VAE) [21, 22] aim to learn a latent space representation of speech, striking a balance between speech quality and compression efficiency. GANs [23, 24] have been explored for speech coding, offering potential for high-quality speech reconstruction and improved modeling of speech parameters.

Encodec [25] and Soundstream [26] are notable neural network-based audio codec methods. Encodec utilizes a hierarchical generative model with VAE for high-fidelity audio coding, while Soundstream combines RNN with an

autoregressive model for efficient audio coding at low bit rates.

Waveform-based methods directly model and reconstruct speech waveforms, achieving high fidelity and naturalness. Parametric-based methods, on the other hand, focus on modeling and coding underlying speech parameters, offering advantages in compression efficiency and manipulation of speech characteristics. However, they may struggle to capture fine details and accurately represent certain speech characteristics, leading to potential artifacts and reduced speech quality.

In general, NN-based speech coding methods have demonstrated remarkable progress, each with its own strengths and limitations. The choice of method depends on specific application requirements, balancing speech quality, compression efficiency, and computational complexity.

Neural network model-based various standards of ADPCM speech coding systems have been developed in recent years [12, 14, 27–31], but they have some limitations. These limitations include the following: (1) simple neural network topologies, (2) lack of neural network training via backpropagation, (3) most studies on neural network-based ADPCM systems were focused on online streaming applications only, and (4) high encoding computational cost due to online learning encoding. Of course, online learning predictive models are able to adapt to new data, which can be useful in situations where data is constantly changing or streaming in dynamic environments. However, compared to the offline learning predictive model, online learning models also have overfitting and lack stability drawbacks [32]. The overfitting, as online models continuously encounter new data, making them more prone to adapting quickly to noise or outliers. This tendency can result in a decrease in the model's ability to generalize well beyond the training data. Additionally, lack stability online learning models, due to their continuous adjustment to new data. Even slight changes in the data distribution can significantly impact the model performance. These limitations, specifically overfitting and lack of stability, can result in a lack of generalization, ultimately impacting the quality and performance of speech coding. Therefore, it is crucial to address these issues to improve the overall quality and performance of speech coding. The high computational cost of encoding for online learning neural network-based ADPCM speech codecs is an additional drawback.

On the other hand, currently, there are several studies on waveform-based recurrent neural network (RNN) prediction models that could be learned from training data by backpropagation, such as [12, 33–38]. Furthermore, the recent RNN architecture-based waveform speech prediction training models [12, 39] and waveform speech generation models [40–42] are clearly and successfully presented. These RNN architecture-based predictor models which were mentioned above are trained using offline stored speech data and offline training approaches.

However, to the best of the researchers' knowledge, no studies have applied or integrated the ADPCM speech coding system based on the RNN architecture predictor model for stored speech (audio) data to improve prediction performance. Considering the gated recurrent unit (GRU) predictor model has the ability to manage both the short-term and long-term predictors of speech sample recall, we are motivated to propose a GRU predictor model-based ADPCM speech coding system for stored speech (audio) data to enhance prediction performance and computational cost.

The rest of the paper is structured as follows. Section 2 describes the methods used in the study. The results and discussion of the experiments were presented in Section 3. Finally, in Section 4, conclusions of the study were presented.

## 2 Methods

In this section, in order to achieve the proposed research objective, the major activities are performed as the following steps. First, data pre-processing was performed to prepare the training and testing speech signal data sets. Second, GRU predictor model was embedded with the ADPCM system. This combined the two technologies, making it possible to effectively train the model. Third, the GRU predictor model was trained using the ADPCM fixed predictor output and the actual PCM speech signal samples. The trained model was then evaluated by test set from the output of ADPCM fixed predictor speech signal samples that was not used for training of the model. Finally, the model that could show high degree of accuracy is integrated with ADPCM decoder system; see the process in Fig. 1.

### 2.1 PCM speech signal dataset

This research explored GRU predictor model-based ADPCM speech coding. A dataset from the DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus [43, 44] was used to train and test the model. This corpus contained 6300 sentences, 10 each from 630 speakers from 8 dialect regions of the USA. Each speech signal dataset was stored as an uncompressed digitized audio file, with a ".wav" file extension and a single-channel signal. Each sentence in the TIMIT dataset was originally sampled at a rate of 16 kHz per second. In this study, according to the objective of the study, the data sets were prepared from the entire TIMIT Acoustic-Phonetic Continuous Speech Corpus. The waveform speech signal data

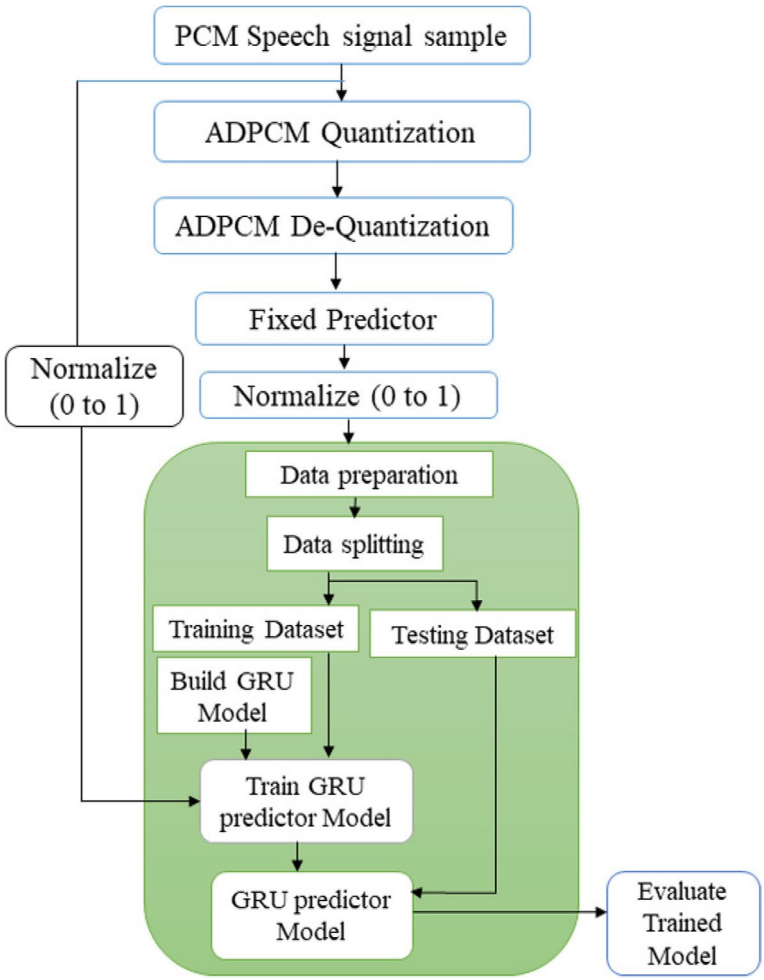


**Fig. 1** Embedding and training of GRU predictor model with ADPCM system

set was broken down into separate training and testing data sets, depicting the percentage distribution in Fig. 2.

We have chosen a total of 40 speakers, comprising 20 females and 20 males. Each speaker produced five sentences, resulting in a combined total of 200 sentences. The training set encompassed 80% of the data, consisting of 32 speakers (16 women and 16 men) and 160 sentences. The remaining 20% of the data formed the test set, which included 8 speakers (4 men and 4 women) and 40 sentences.

The duration of the speech signals used for training amounted to 526 s, equivalent to 8 min and 46 s. The overall duration of the testing signals was 136 s or 2 min and 16 s. The TIMIT Acoustic Phonetic Continuous Speech Corpus was digitized using waveform speech with a uniform PCM sampling rate of 16,000 samples per second. Consequently, the training dataset comprised 8,443,764 samples, while the testing dataset consisted of 2,187,074 samples.

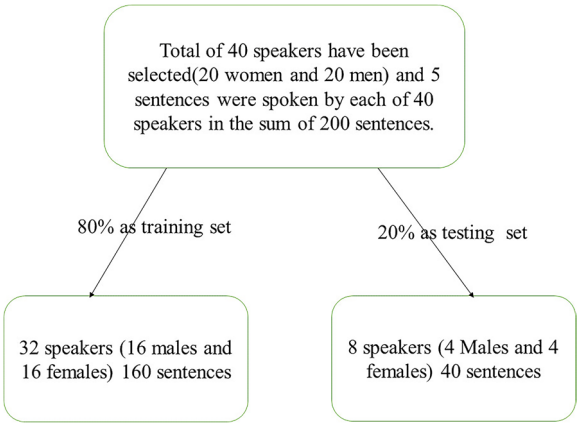


**Fig. 2** Training and testing dataset percentage distribution ratio

### 2.2 Waveform-based speech signal predictive analysis

Consider a speech signal with $x(t)$, $x(t\text{-}1)$, $x(t\text{-}2)$,..., $x(t\text{-}n)$ samples. In the waveform-based speech prediction process, the current speech sample is estimated as a linear or nonlinear function of a fixed number of previous consecutive speech samples, that is, the prediction of a speech sample at time $t$ is as shown in Eq. 1:

$$\hat{x}(t) = f(x(t-1), x(t-2), x(t-3), \ldots, x(t-p)) \ldots\ldots \quad (1)$$

where $f$ is the linear or nonlinear function used for prediction, and $p$ is the number of previous consecutive speech samples used in the prediction.

A discrete signal $x(t)$ is simply a sequence of numbers corresponding to the signal samples, which is sampled uniformly at an arbitrary sampling rate (shown in Fig. 3). The usual sampling rate for speech recognition applications is 16 kHz. This is essentially the information we find in a PCM-encoded WAV file.

Speech is continuous signal, which means that consecutive samples of the signal are correlated (see Fig. 3). So, if we have the previous sample, $x(t\text{-}1)$, then we can predict the next sample, $\hat{x}(t)$, based on the previous sample, $x(t\text{-}1)$, and it should be roughly equivalent to $x(t)$. Furthermore, if we utilize more prior samples, we can gain additional data, which will help us make a better prediction. Furthermore, if we utilize more prior samples, we can gain additional data which will assist in making a better prediction. Explicitly, we can set up a predictor that utilizes $P$ prior samples to predict the present sample $x(t)$, as demonstrated in Fig. 4.

### 2.3 Proposed GRU predictive model with ADPCM speech coding architecture

In this proposed model, the GRU predictive model is integrated with the ADPCM speech encoder system to train the proposed system. During the training phase, the GRU model is fed with the PCM speech samples, $x(t)$ and the corresponding ADPCM fixed predictor output, $\bar{x}(t)$, shown in Fig. 5. It learns to predict future values based on this input data. Once the GRU model is trained, it is saved and evaluated to assess its performance and accuracy in predicting future speech samples. After training

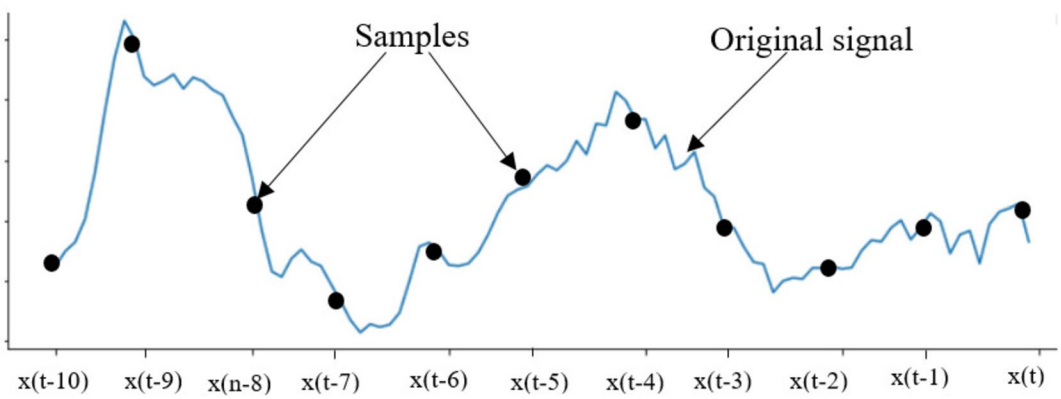


**Fig. 3** Sequences of samples representing speech signal

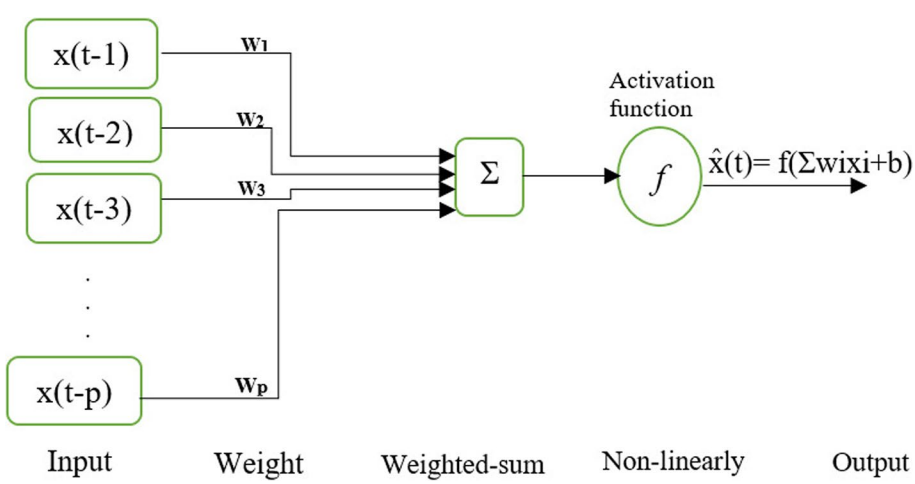


**Fig. 4** A neuron network employing non-linearity and weighted sum of previous components to predict the current one

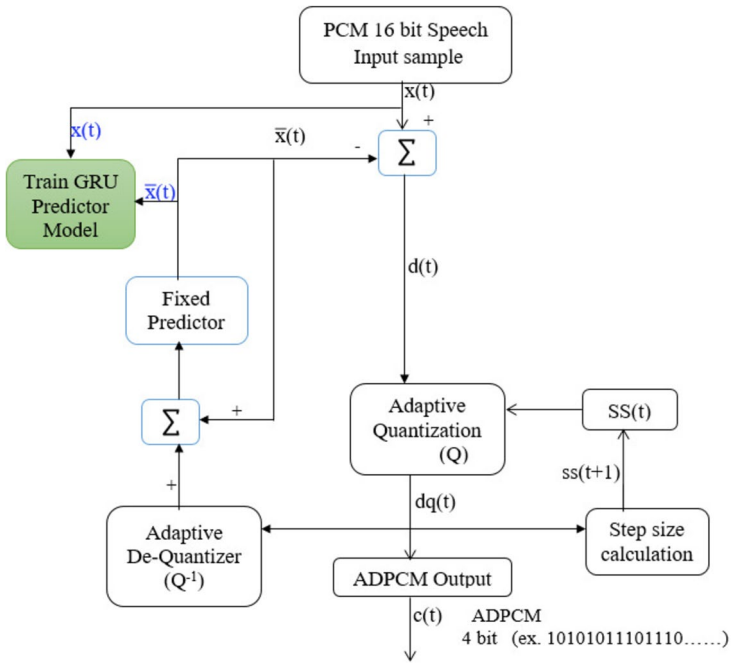


**Fig. 5** GRU predictive model with ADPCM speech encoder training process

the GRU model, the conventional ADPCM encoder is continued without the GRU predictor. And then, the trained GRU model should be saved and evaluated. The trained GRU model is deployed in the ADPCM speech decoder system, as illustrated in Fig. 6. The decoder system utilizes the GRU model's predictive capabilities to reconstruct the original speech samples from the ADPCM-encoded data. The goal of this integration is to train a more effective GRU predictive model by incorporating the ADPCM speech encoder system and utilizing it for training the ADPCM speech decoder system.

#### *2.3.1 GRU predictive model training with ADPCM system*

Our study presents a novel approach to integrating the GRU predictive model with the ADPCM codec for speech signal prediction. Our contribution includes the development of a training algorithm for the GRU predictive model that uses a data set of original PCM speech samples and the ADPCM fixed predictor output, represented as a set of $x(t)$ and $\bar{x}(t)$. The dataset is normalized and modified to be compatible with the GRU predictive model, with input in the format of $(\bar{x}_{\text{train}})$ and $(x_{\text{train}})$ for training the model, as shown in Table 1. Additionally, we utilize the GRU gate, which controls the flow of information between the current and previous time steps in the GRU neural network, composed of the update and reset gates, to improve the accuracy of the model. In general, our work presents a unique application of the GRU predictive model and the ADPCM codec in the prediction of speech signals.

The equations of (2) and (3) for this GRU predictive would be as follows.

$$\bar{x}(t) = f(\bar{x}(t-1), \bar{x}(t-2), \ldots, \bar{x}(t-p))$$

Ideally, the reconstructed speech sample should closely look like the original speech sample.

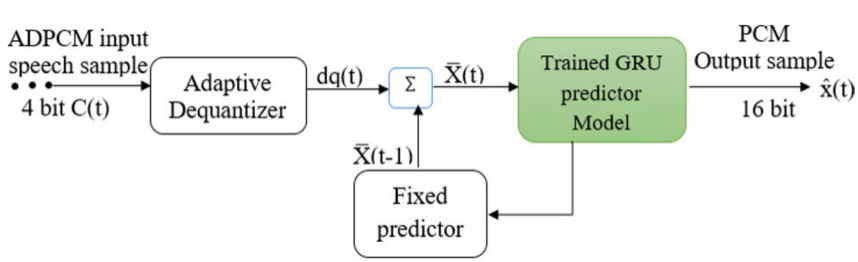


**Fig. 6** GRU predictive model-based ADPCM decoder system

**Table 1** GRU predictive model sample input and output dimensions for model training

| $\bar{x}(t)$ | $x(t)$ |
|---|---|
| $\bar{x}(t-1), \bar{x}(t-2), \bar{x}(t-3), \ldots, \bar{x}(t-p)$ | $x(p)$ |
| $\bar{x}(t-2), \bar{x}(t-3), \bar{x}(t-4), \ldots, \bar{x}(t-(p+1))$ | $x(p+1)$ |
| $\bar{x}(t-3), \bar{x}(t-4), \bar{x}(t-5), \ldots, \bar{x}(t-(p+2))$ | $x(p+2)$ |
| $\bar{x}(t-4), \bar{x}(t-5), \bar{x}(t-6), \ldots, \bar{x}(t-(p+3))$ | $x(p+3)$ |
| ... | ... |

$$\bar{x}(t) \approx x(t) \tag{2}$$

$$\bar{x}(t) = f(\bar{x}(t-1), \bar{x}(t-2), \ldots, \bar{x}(t-p)) \approx x(t) \tag{3}$$

where $f$ represents the function of the GRU predictive model; the model takes a sequence of previous ADPCM fixed predictor speech samples ($\bar{x}(t)$) and predicts the next original speech sample ($x(t)$). The input dimensions are $p$ by 1, and the output dimensions are 1 by 1, as shown in Table 1.

#### *2.3.2 GRU structure and speech signal prediction*

The GRU architecture allows for the preservation of information from earlier parts of a sequence while also being able to handle dependencies in large data sequences. This is achieved by using gates that determine which information to keep or discard at each time step. Our analysis revealed how previous speech samples are processed within the gates and how the current sample is predicted, as illustrated in Fig. 7.

Update gate

The update gate is responsible for collecting data that can be used to determine the amount and type of information required.

$$z_t = \sigma(W_z \cdot \bar{x}(t) + U_z \cdot h_{t-1}) \tag{4}$$

The current input speech sample, $\bar{x}(t)$, is multiplied with its update weight input, $W_z$, and added to the multiplication of the previous hidden state, $h_{t-1}$, and the previous hidden weight, $U_z$. Both outputs are processed together and multiplied by a sigmoid function, $\sigma$, which produces the update gate $z_t$, with a value between 0 and 1.

Reset gate

The input sample, $x(t)$, is multiplied by the reset weight, $W_r$, and added to the previous hidden state, $h_{t-1}$ and the reset weight of the hidden state, $U_r$, in the reset gate. Then, a sigmoid function is applied to scale the output between 0 and 1.

$$r_t = \sigma(W_r \cdot \bar{x}(t) + U_r \cdot h_{t-1}) \tag{5}$$

The reset gate determines the amount of data to be erased, along with the unwanted data being reset.

$$h'_t = \tanh(W \cdot \bar{x}(t) + r_t \cdot Uh_{t-1}) \tag{6}$$

The important information that needs to be remembered is conveyed to the reset gate through $h_t$ which is the

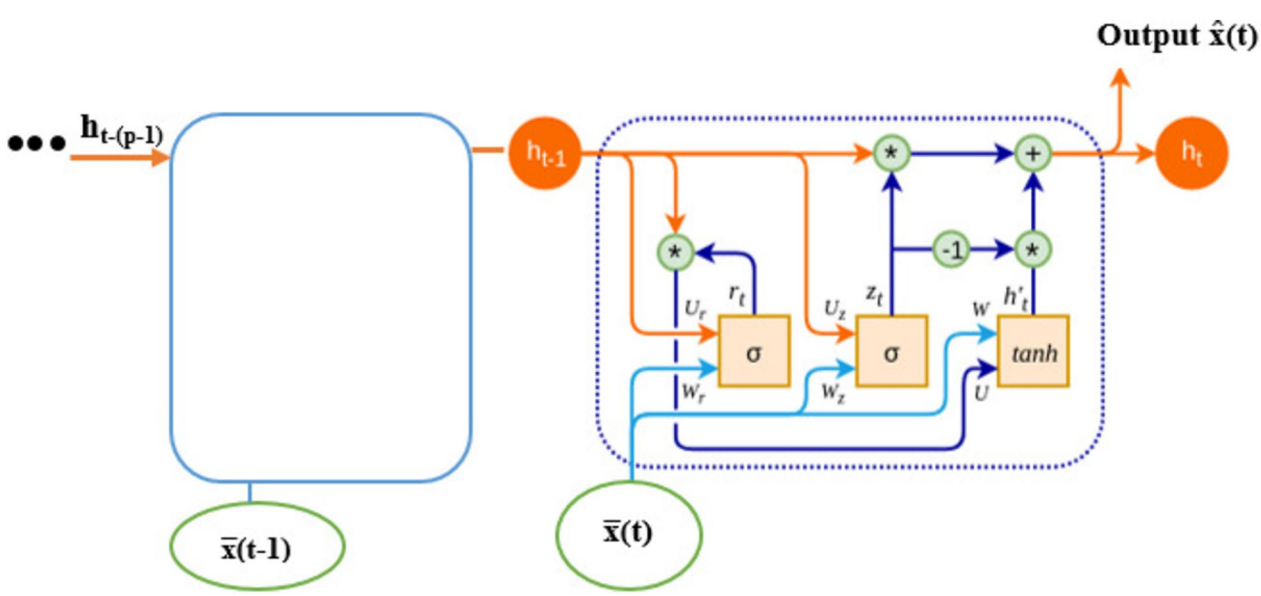


**Fig. 7** Structure of the GRU speech signal predictive model

following: the result of $z_t$ and $h_{t-1}$ was element wise added, and the result of $(1 - z_t)$ and $h'_t$ was element-wise added as well.

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot h'_t \tag{7}$$

After being trained according to Table 1 and using the predictive model of GRU with ADPCM, to predict the current sample based on the previous context, the GRU passes the previous context through a set of distinct gates. The reset gate helps to make the network forget outdated information while the update gate helps the network to remember important information. The input gate then allows the network to update its memory based on the current input. Finally, the output gate allows the network to generate the predicted output. Thus, $h_t$ is the predicted GRU output sample.

### 2.4 GRU predictive model evaluation

After the GRU predictive model has been trained using the ADPCM encoder, it should be evaluated by running test sets before being deployed on the ADPCM decoder side.

In this study, the signal-to-noise ratio (SNR) method was used to evaluate the GRU predictor model of speech coding quality. SNR [45, 46] was used as a measure to determine the quality of the predicted speech. In this measure, original PCM speech signal and the predicted speech signal predicted by the GRU with ADPCM system were compared to identify any differences, which would indicate the presence of noise.

To determine SNR, consider a collection of real speech signal samples, $x(t)$, and a predicted noise signal, $\hat{x}(t)$. The difference between $x(t)$ and $\hat{x}(t)$ is known as the error $e(t)$, which encompasses both noise and distortion. The objective is to compute the ratio of the power of the signal to the power of the noise in order to estimate the quality of the signal.

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{8}$$

We are now able to calculate the power level of the speech signals, measured in decibel(dB), referred to as $P_{signal}$.

$$P_s = 10 \log_{10}(P_{signal}) \tag{9}$$

To find the noise, we first calculate the power of the actual input speech signal, referred to as $P_i$, and the power of the predicted speech signal, referred to as $P_o$. The difference between these two values is considered to be the noise level, represented by $P_n$.

$$P_n = 10 \log_{10}(P_{noise}) \tag{10}$$

### 2.5 Integrating and training GRU predictor model with ADPCM speech coding algorithm

The proposed method for using a GRU predictor model with an ADPCM system for speech signal coding processing step by step summery is the following.

Step 1: Identified and imported all required libraries

Step 2: Load PCM actual speech sample and output of ADPCM fixed predictor speech sample

```
Load: pcm_sample and adpcm_samples
pcm_samples = load_pcm_samples()
adpcm_samples = load_adpcm_samples()
```

Step 3: Normalize the speech samples from range (-32,768 to 32,767) into (0 to 1)

```
pcm_normalized = norm_samples
                      (pcm_samples)
adpcm_normalized = norm_samples
                      (adpcm_samples)
```

Step 4: Define the input and output sequences for the GRU-based predictor model

```
samp = 10 # Number of past samples
steps = 1 # Number of future steps
X = []  #X array
Y = []  #Y array
for i in range(pcm_normalized.shape[0]
                              - samp):
X.append(adpcm_normalized[i:i+samples])
Y.append(pcm_normalized[i+samples-1])
```

Step 5: Training the model with the given inputs

```
history = model.fit(X[:], Y[:],
                    epochs=50,
                    batch_size=32,
                validation_split=0.2,
                verbose=1)
```

Step 6: Define the GRU-based predictor model architecture

```
model = keras.Sequential()
# Add a GRU layer with 3 units.
model.add(
 layers.GRU(3,
  activation = "tanh",
  recurrent_activation = "sigmoid",
  input_shape=(X.shape[1],X.shape[2])))
model.add(layers.Dropout(rate=0.2))
model.add(layers.Dense(1))
```

Step 7: Compile trained GRU with ADPCM model with optimizer as Adam and loss as MSE

```
model.compile(loss='mean_squared_error',
                optimizer = 'adam')
```

Step 8: Save the trained GRU predictor model with the ADPCM encoder

```
model.save("gru_adpcm_model.h5")]
```

Step 9: Preparing testing set

```
testX, testY =  X[:], Y[:]
```

Step 10: One sample prediction per step head

```
y_pred = loaded_model.predict(testX)
```

Step 11: Evaluate the model using SNR

```
GRU_Error_Signal = y_pred - testY
dnn_mse = 10*np.log10(
      np.mean(pow(np.abs(
      GRU_Error_Signal [:]),2)))
dnn_sigpow = 10*np.log10(
   np.mean(pow(np.abs(testY),2)))
GRU predictor SNR = dnn_sigpow
                - dnn_mse, "dB")
```

Step 12: Deployed/integrated trained GRU predictive model with ADPCM speech decoder

## 3 Results and discussion

In this section, we report the results of our experimental study on the embedded GRU predictive model-based ADPCM decoder system. The experiments were carried out using the Python 3.9.7 Anaconda platform on a Jupiter notebook, TensorFlow version 2.9.1, and Keras 2.6.0 on a Windows 10 Pro, 22H2 operating system. The computer used had an Intel® CoreTM i5 6th generation processor, with a speed of 2.30 GHz, 2.40 GHz, and 16 GB of RAM.

In this study, four different experimental settings were examined. The first used the baseline of the IMA ADPCM speech codec with a fixed predictor. The second configuration used of an online training RNN predictor with the IMA ADPCM speech codec. The third experiment was online learning GRU predictor model with IMA-ADPCM speech codec. Lastly, the fourth configuration employed trained GRU predictive model, embedded with the IMA ADPCM speech decoder.

To properly evaluate each predictor's error (specifically experiments 1, 2, and 3 compare with GRU batch predictor model), we need to divide the test data set and demonstrate the evaluation in three steps. This is necessary for the experiment 2 and 3 to have enough time to converge when measuring the mean square error and the SNR. The total sample length of the speech signal of the test set is 2,187,074 samples. Each predictor will be evaluated in four stages as the following setting:

- Experiment $i$, first stage: 1/3 of the first test set speech signal samples (0 to 729,000)
- Experiment $i$, second stage: 1/3 of the middle test set speech signal samples (729,001 to 1,458,050)
- Experiment $i$, third stage: 1/3 of the last test set speech signal samples (1,458,051 to 2,187,074)
- Experiment $i$, fourth stage: the whole test set speech signal samples (0 to 2,187,074)

where $i = 1, 2, 3$, and 4 experiments.

### 3.1 Experiment 1: ADPCM speech CODEC with fixed predictor

In experiment 1, we used a fixed predictor ADPCM codec. This codec utilizes adaptive quantization and a fixed prediction process. The encoding process involves converting a 16-bit PCM sample, $x(t)$, into a 4-bit ADPCM sample, $c(t)$. This compressed 4-bit ADPCM data can be used for transmission in low band and stored on the disk. The compressed data is then reconstructed into the original signal.

In the decoding process, the ADPCM decoder takes in the 4-bit code from the encoder output and generates a 16-bit predicted speech signal sample. To produce a new predicted difference value, the 4-bit ADPCM code input is inverse-quantized. The new value of the speech sample is calculated by adding this value to the previous predicted value of the speech sample. The new step size

index is obtained by adding the value of modifications to the present index. The final result is a new 16-bit sample that is reconstructed.

The quality of the reconstructed signal was evaluated against the original signal using the SNR, which was recorded in 31.5 dB throughout all evaluation stages.

### 3.2 Experiment 2: ADPCM with online learning RNN predictor

A simple recurrent neural network (RNN) model can be integrated with the ADPCM speech coding system to improve its predictive capabilities. The RNN model takes a short history of previous speech samples and predicts the next sample. This predicted sample is then used by the ADPCM encoder instead of the original predicted sample. The RNN model architecture consists of an input layer to receive the previous speech samples, a hidden recurrent layer to capture temporal context, and an output layer to predict the next sample. The network is trained on speech data to minimize the prediction error using backpropagation over time. The online RNN model is then integrated into the ADPCM system by feeding the RNN predicted sample into the quantizer and encoder instead of the prediction from the original ADPCM algorithm. The experiment shows the integrated online RNN-ADPCM system provides improved speech quality and lower distortion compared to the baseline ADPCM fixed predictor coder alone.

Figure 8 depicts the experimental result of an ADPCM codec based on an online RNN predictor model. In the first stage of experiment 2, a subset of speech signal samples from the initial test set (ranging from 0 to 729,000) was used to evaluate the performance of ADPCM with the RNN predictor. The resulting SNR was 36.4 dB. In the second and third phases of experiment 2, the test set of speech signal samples ranging from 729,001 to 1,458,050 and 1,458,051 to 2,187,074 have been used to evaluate the RNN predictor model-based ADPCM codec. The obtained results were 40.3 dB and 42.5 dB respectively. In the fourth stage of experiment 2, all speech signal samples (from 0 to 2,187,074) were used to evaluate the model, resulting in SNR of 39.7 dB. Table 2 presents the results of experiments 1, 2, 3, and 4.

Figure 8a displays the actual, predicted, and error signals of the ADPCM codec with the online RNN predictor model on the entire test set. Furthermore, Fig. 8b shows 100 randomly selected samples from the range of 300,000 to 300,100, and Fig. 8c provides a zoomed-out

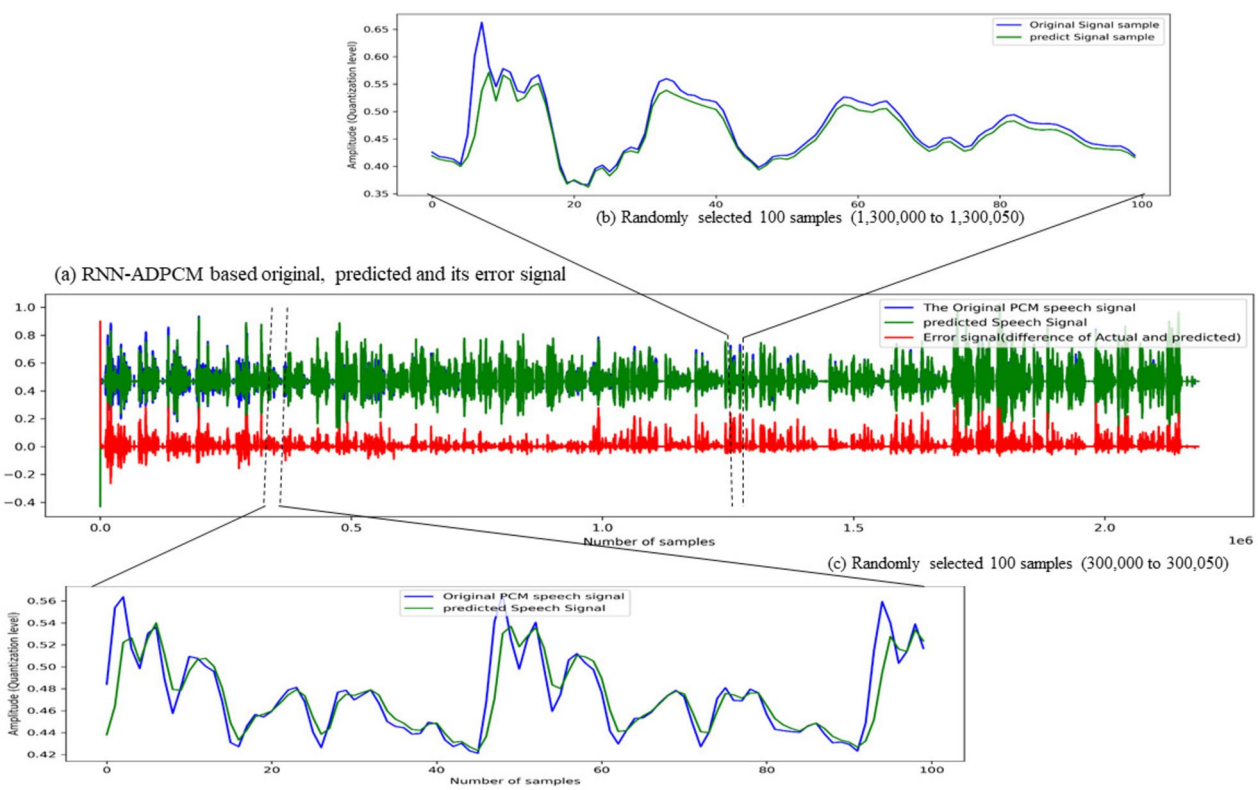


**Fig. 8** **a** The actual, predicted, and error signals of the IMA-ADPCM codec with the online RNN predictor model applied to the entire test set. **b**, **c** 100 randomly selected samples from specific ranges, offering insights into the performance of the model in capturing and predicting speech signals

**Table 2** The four experimental SNR results in various speech signal data test sets

| Test-set is divided into 4 stages | EXP1 | EXP2 | EXP3 | EXP4 |
|---|---|---|---|---|
| 1st stage samples (0 to 729,000) | 31.6 | 36.4 | 36.8 | 44.6 |
| 2nd stage samples (729,001 to 1,458,050) | 31 | 40.3 | 40.5 | 45 |
| 3rd stage samples (1,458,051 to 2,187,074) | 32.2 | 42.5 | 42.6 | 44.7 |
| The whole test set samples (0 to 2,187,074) | 30.9 | 39.7 | 40.1 | 44.7 |

view of the predictive error for 100 randomly selected samples ranging from 1,300,000 to 1,300,100 to visualize the difference of actual and predicted signal.

### 3.3 Experiment 3: online learning GRU predictor model with IMA-ADPCM speech codec

Experiment 3 extends the approach of experiment 2 that integrated GRU predictor model with the ADPCM speech coding system. The main difference between experiments 2 and 3 (RNN- and GRU-based ADPCM codec) lies in their internal architecture. RNN has a simple structure that passes information from one step to the next, but they struggle with capturing long-term dependencies due to vanishing or exploding gradient problems. GRU, on the other hand, incorporates gating mechanisms that enable better handling of long-range dependencies by selectively updating information. GRU has a more sophisticated design with reset and update gates, allowing them to effectively capture relevant information over longer sequences while mitigating some of the challenges associated with RNNs.

Both experiments adopt an online learning strategy, with the predictor model trained to minimize quantization errors. Results indicate improved speech quality and reduced distortion compared to the baseline ADPCM fixed predictor and online learning RNN predictor-based ADPCM codec.

The experimental results, depicted in Fig. 11 and summarized in Table 2, showcased the effectiveness of the online GRU predictor model across different phases of testing. In the initial stage, utilizing a subset of speech signal samples, the SNR was measured at 36.8 dB. As the evaluation extended to larger portions of the test set in subsequent phases, SNR values of 40.5 dB and 42.6 dB were achieved. In the final stage, encompassing the entire test set, the model yielded an SNR of 40.1 dB.

The integration of online learning models with ADPCM speech coding systems can be enhanced by examining the particular details in performance and predictive abilities, even if both experiments demonstrated enhancements in speech quality and reduction of distortion.

### 3.4 Experiment 4: GRU predictor model-based ADPCM speech decoder

In this study, we integrated the GRU prediction model with the ADPCM codec to enhance the encoding quality. By utilizing the Adam optimization technique and a batch size of 32, GRU prediction model is trained using 50 epochs, while incorporating various amounts of previous sample sizes.

In addition, we examined various numbers of previous speech sample sizes to predict the current sample. The correlation function of previous speech samples can be used to describe the dynamic GRU prediction model. According to the experimentation, the use of the number of previous speech sample sizes such as 3, 5, 7, 10, 12, and 15 resulted in predicted SNR accuracy values ranging from 38.9 to 41.5 dB, as depicted in Fig. 9,when using a sample size of 10, outperforms the better performance of other in terms of prediction accuracy.

In the fourth experiment's first stage of testing, the trained GRU predictive model-based IMA-ADPCM decoder was evaluated by a portion of the test set consisting of 729,000 speech signal samples in range from 0 to 729,000. We then could obtain a 44.6 dB result. In the second stage of experiment 3, we used to evaluate the model 1/3 of the entire middle range of the test set speech samples, ranging from 729,001 to 1,458,050. The GRU predictor model resulted in an SNR of 45 dB. The third stage of experiment 4 involved to evaluate the proposed model using the last portion of 1/3 of the entire speech signal sample test set in between 1,458,051 and 2,187,074. The result obtained from SNR was 44.7 dB. And lastly, in the fourth stage of this experiment, all speech signal samples (from 0 to 2,187,074) were used to evaluate the model, resulting in an SNR of 44.8 dB.

The experimental results of the GRU predictor model-based ADPCM speech coding actual, predicted, and quantization error speech signals are shown in Fig. 10.

In order to visualize, the experimental plotted results of quantization error (difference) of actual and predicted signal are depicted in Fig. 10a–c. Figure 10b shows 100 randomly selected samples from the range of 300,000th to 300,100th, and Fig. 10c provides a zoomed-out view of the predictive error for 100 randomly selected samples ranging from 1,300,000 to 1,300,100.

Furthermore, Figs. 8 and 10 illustrated the visual performance differences between the online RNN predictive model-based ADPCM codec and the proposed GRU predictive model-based ADPCM decoder.

Table 2 illustrates the SNR coding for different sets of speech signal test set using ADPCM with a fixed predictor, ADPCM with an online RNN predictor, and ADPCM with GRU predictor model. Figure 11 compares the results of the four experimental settings

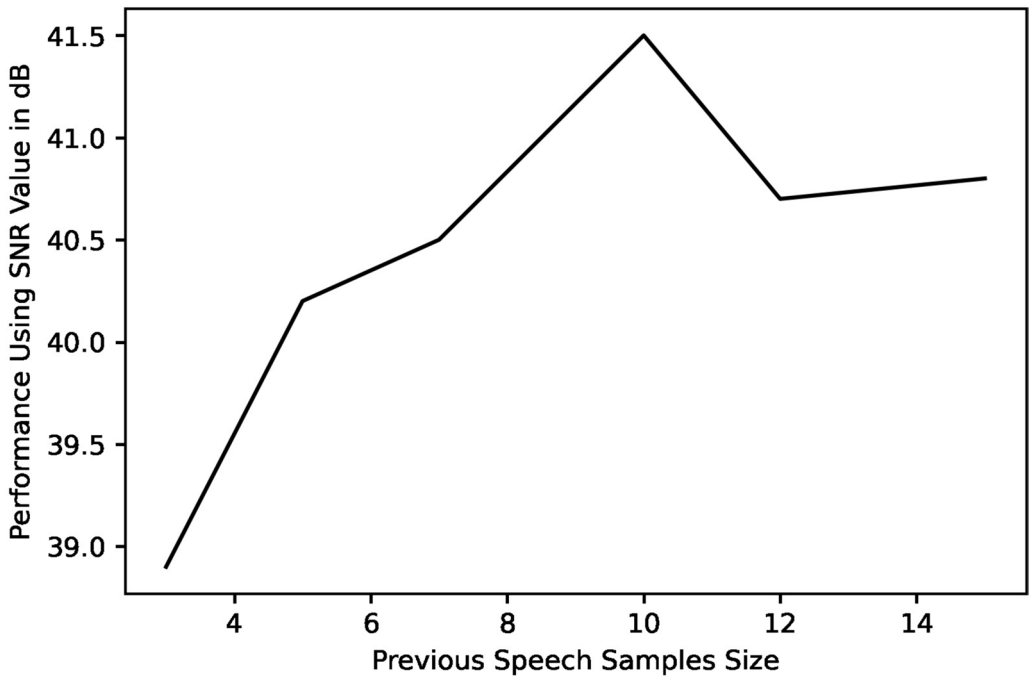


**Fig. 9** ADPCM with GRU predictor model SNR value of performance in different previous sample size

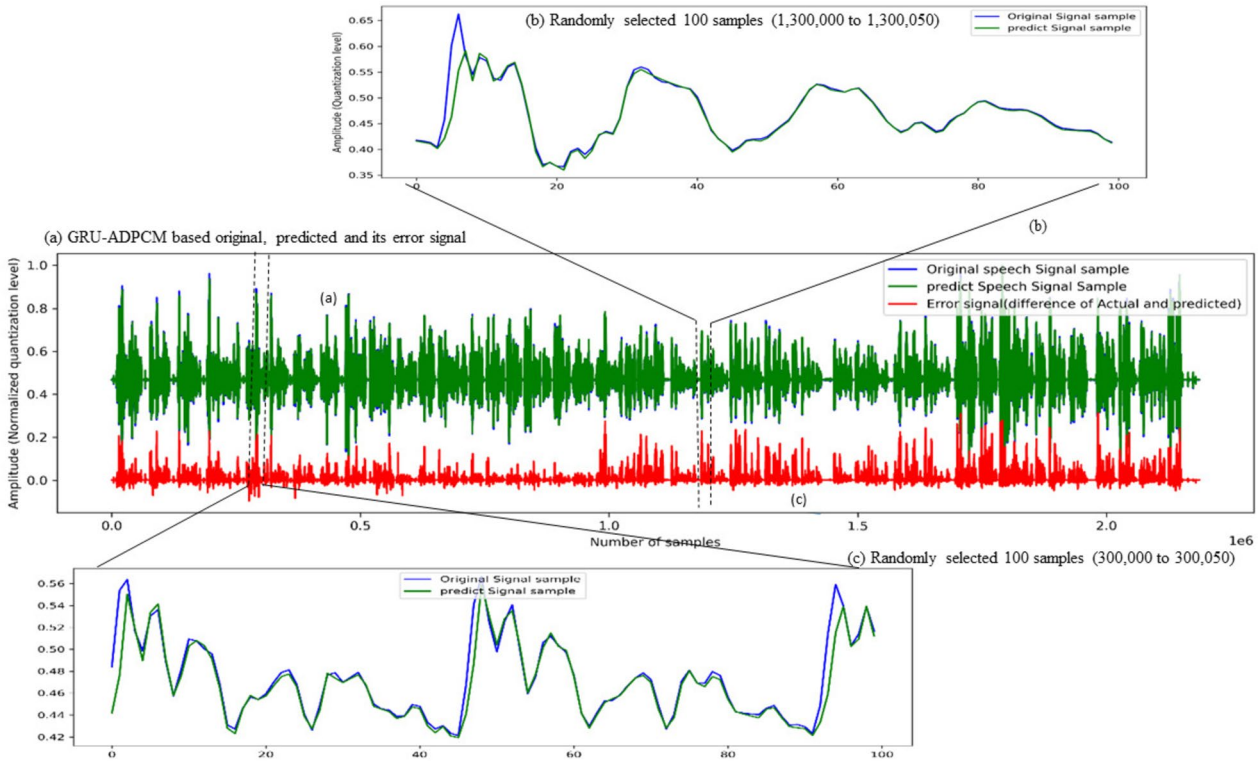


**Fig. 10** GRU predictor model based on ADPCM coding of actual, predicted, and quantization error speech signals

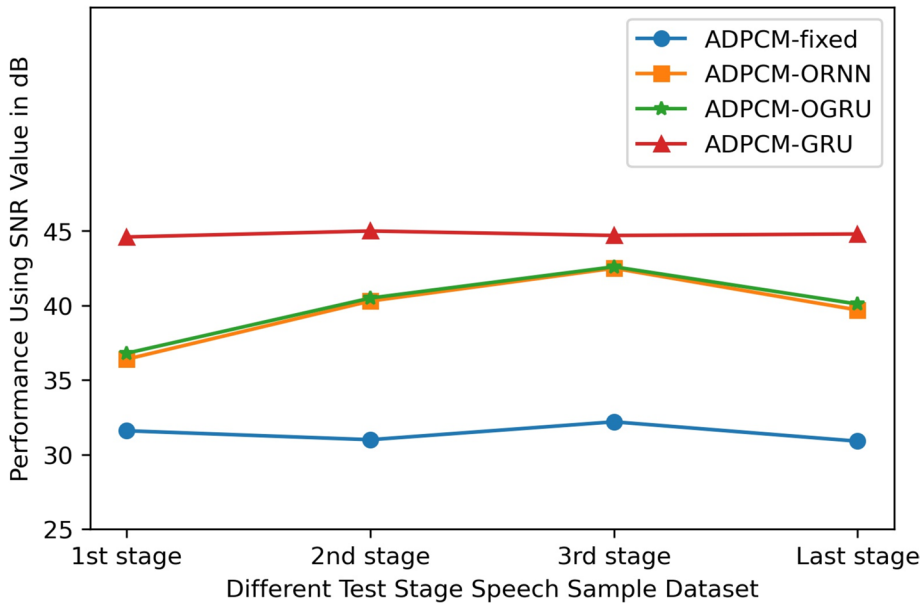


**Fig. 11** Comparison of ADPCM predictor performance and experimental results

using a different stage of the test set: ADPCM with fixed predictor, ADPCM with online RNN train predictor, and ADPCM with the GRU predictor model. The test results are shown in four stages: the first stage test set (0 to 729,000 samples), the second stage test set (729,001 to 1,458,050), the third stage test set (1,458,051 to 2,187,074), and the last stage whole test set (0 to 2,187,074).

As shown in Fig. 11, the ADPCM-fixed predictor setting has relatively stable test results at all stages, with values close to 31.5. In the ADPCM online RNN and GRU predictor setting, it has higher results in the second and third stages, which adaptable incremental enhancing, but in the last stage (average of all stages), a lower result which is used for the whole data set. The ADPCM-GRU predictor model setting has the highest test results, with values close to stable to 45 at all stages.

The proposed GRU predictor model for IMA-ADPCM decoding stands out as the high-quality among the examined predictors for several key reasons. Firstly, the proposed model trained by leveraging the fixed predictor's output and the actual speech sample data separately in encoder side by back propagation could optimize the Wight. This integration allows the GRU predictor to continuously refine its predictions based on the actual speech samples, resulting in enhanced decoding quality, as evident from the consistently higher SNR values across all test stages.

Additionally, the proposed GRU predictor model excels in terms of computational efficiency, as highlighted in experiment 4. Unlike the online RNN and GRU predictors in experiments 2 and 3, respectively, the proposed model is trained during a separate training phase using a large dataset. Once trained, the model is saved and deployed on the decoder part, eliminating the need for continuous computations during encoding in real-time applications. This unique approach significantly reduces the encoding computational cost, offering a balanced solution that prioritizes predictive accuracy while mitigating processing speed concerns. In contrast, the online RNN and GRU predictors incur higher computational costs as they compute predictions for each sample during both encoding and decoding processes.

In summary, the proposed GRU predictor model emerges as the preferred choice due to its ability to enhance predictions through a hybrid learning approach and its efficiency in terms of encoding computational cost, addressing key challenges associated with real-time applications and showcasing superior performance in decoding quality.

## 4 Conclusions

We proposed a GRU predictor model-based IMA-ADPCM speech decoder system for to enhance prediction performance and computational encoding cost.

In this study, four experiments have been examined: the baseline or fixed predictor-based IMA-ADPCM speech codec, online learning RNN predictor-based IMA-ADPCM, online learning GRU predictor-based IMA-ADPCM, and GRU predictor model IMA-ADPCM decoder.

The results of the experiment show that the GRU predictor model-based ADPCM decoder had the highest SNR, indicating that it was the most accurate in predicting the speech signals. The online RNN and GRU predictor also improved the predictive ability of IMA-ADPCM coding over time eventually, but it was not performed the same as the proposed model.

This proposed integrated GRU predictor with IMA-ADPCM decoder significantly improves the accuracy of speech predictions, making it a promising approach for speech coding applications. The proposed model also could remove the online learning-based predictors' encoding computational cost.

Our contribution includes the development of an algorithm to train the GRU model using a data set of PCM speech samples and the ADPCM fixed predictor output. Due to this, the study improves the decoding performance and decreases the encoding computational cost.

Although the proposed GRU predictor model with ADPCM coding shows promising results in terms of speech quality and prediction accuracy, its computational cost and complexity need further study. Thus, future research should assess the model's computational efficiency and complexity of the model and evaluate its performance with other speech coding metrics. Furthermore, it would be interesting to investigate the integration of the proposed model with other speech coding techniques such as parameter coding and vector quantization, to further improve its performance. Additionally, it would be beneficial to investigate the proposed model's performance in different languages and under different noise conditions.

## Appendix

### Some sample Python codes

#### *IMA ADPCM encoding (16-bit PCM sample to 4-bit ADPCM sample word length) Python sample code*

IndexTable = [-1, -1, -1, -1, -1, 2, 4, 6, 8, -1, -1, -1, -1, 2, 4, 6, 8]

StepSizeTable = [7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 19, 21, 23, 25, 28, 31, 34, 37, 41, 45, 50, 55, 60, 66, 73, 80, 88, 97, 107, 118, 130, 143, 157, 173, 190, 209, 230, 253, 279, 307, 337, 371, 408, 449, 494, 544, 598, 658, 724, 796, 876, 963, 1060, 1166, 1282, 1411, 1552, 1707, 1878, 2066, 2272, 2499, 2749, 3024, 3327, 3660, 4026, 4428, 4871, 5358, 5894, 6484, 7132, 7845, 8630, 9493, 10442, 11487, 12635, 13899, 15289, 16818, 18500, 20350, 22385, 24623, 27086, 29794, 32767]

```
# ADPCM_Encode.
# sample: a 16-bit PCM sample
# retval : a 4-bit ADPCM sample
predsample = 0
index = 0

# Encoding from PCM to ADPCM defined function
def ADPCM_Encode(sample):
    global index
    global predsample
    global diffq
    global ore_diff
    global error_signal

    code = 0

    step_size = StepSizeTable[index]

    # compute diff and record sign and absolut value

    diff = sample - predsample
    ore_diff=diff
    #print('diff sample:' ,ore_diff)
    if diff < 0:
        code = 8
        diff = -diff

    # quantize the diff into ADPCM code
    # inverse quantize the code into a predicted diff
    tmpstep = step_size
    diffq = step_size >> 3
    if diff >= tmpstep:
        code = code | 0x04
        diff -= tmpstep
        diffq = diffq + step_size

    tmpstep = tmpstep >> 1
    if diff >= tmpstep:
        code = code | 0x02
        diff = diff - tmpstep
        diffq = diffq + (step_size >> 1)

    tmpstep = tmpstep >> 1
    if diff >= tmpstep:
        code = code | 0x01
        diffq = diffq + (step_size >> 2)

    # fixed predictor to get new predicted sample
    if code & 8:
        predsample = predsample - diffq
    else:
        predsample = predsample + diffq

    # check for overflow
    if predsample > 32767:
        predsample = 32767
    elif predsample < -32768:
        predsample = -32768
    # find new stepsize index
    index += IndexTable[code]
    # check for overflow
    if index < 0:
        index = 0
    if index > 88:
        index = 88
    error_signal = sample - predsample
    # return new ADPCM code code & 0x0f == code
    return code & 0x0f
```

### Decoding from ADPCM to PCM ( a 4-bit ADPCM sample to 16-bit PCM)

```
# ADPCM_Decode.
# code: a byte containing a 4-bit ADPCM sample.
# retval: 16-bit ADPCM sample
de_index = 0
de_predsample = 0

def ADPCM_Decode(code):
    global de_index
    global de_predsample
    step_size = StepSizeTable[de_index]
    # inverse code into diff
    diffq = step_size >> 3  # == step/8
    if code & 4:
        diffq += step_size
    if code & 2:
        diffq += step_size >> 1
    if code & 1:
        diffq += step_size >> 2
    # add diff to predicted sample
    if code & 8:
        diffq = -diffq
    de_predsample += diffq
    # check for overflow  clip the values
```

```
# to +/- 2^11 (supposed to be 16 bits)
if de_predsample > 32767:
    de_predsample = 32767
elif de_predsample < -32768:
    de_predsample = -32768
# find new quantizer step size
de_index += IndexTable[code]
# check for overflow
if de_index < 0:
    de_index = 0
if de_index > 88:
    de_index = 88
# save predict sample and de_index for next iteration
# return new decoded sample
# The original algorithm turned out to be 16bits,
   # need to convert to 16bit
return de_predsample
```

### Load speech signal sample data to encoding and decoding

```
import matplotlib.pyplot as plt
import IPython.display as ipd
import numpy as np
import wave
from scipy.io import wavfile

audio_file = 'audio_data/Train_all_dataset.wav'
fs, data = wavfile.read(audio_file)
wavdata=data[::]

org_samp=[]
pre_samp=[]
err_signal=[]

for i in range(len(wavdata)):
    ADPCM_sample = ADPCM_Encode(wavdata[i])
    pred_sample = ADPCM_Decode(ADPCM_sample)
    org_samp.append(wavdata[i])
    pre_samp.append(pred_sample)
org_samp=np.array(org_samp) #oreginal signal sam.
pre_samp=np.array(pre_samp) #predicted sample
e_signal=np.array(err_signal) #the error of pred.
```

### Normalize data between 0 and 1

```
y = org_samp
ypn = pre_samp
org_samp = np.array(y)
# goal : range [0, 1]
org_samp = (org_samp - min(org_samp)) /
            ( max(org_samp) - min(org_samp) )
pre_samp = np.array(ypn)
pre_samp = (pre_samp - min(pre_samp)) /
            ( max(pre_samp) - min(pre_samp) )
```

### The input data preparation for GRU predictive model training

```
# Need the data to be in the form such as sample, time steps,
samples = 10      # Number of samples (in past)
steps = 1         # Number of steps (in future)
X = [] # X array
Y = [] # Y array
for i in range(wavdata.shape[0] - samples):
    X.append(pre_samp[i:i+samples])     # Independent Samples
    Y.append(org_samp[i+samples-1])    # Dependent Samples
X = np.array(X)
X = np.expand_dims(X, axis=2)
Y = np.array(Y)
Y = np.expand_dims(Y, axis=1)

print('Training Data: Length is ',len(X[0:1][0]),': ', X[0:1])
print('Testing Data: Length is ', len(Y[0:1]),': ', Y[0:1])
print('Dimensions of X:', X.shape)
print('Dimensions of Y:', Y.shape)
```

### Build GRU model architecture

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
model = keras.Sequential()
# Add a GRU layer with 3 units.
model.add(layers.GRU(3,
                     activation = "tanh",
                     recurrent_activation = "sigmoid",
                     input_shape=(X.shape[1], X.shape[2])))
# Add a dropout layer (penalizing more complex models)
model.add(layers.Dropout(rate=0.2))
# Add Dense layer with 1 units (Since doing a regression task.
model.add(layers.Dense(1))

# Evaluating loss function of MSE using the adam optimizer.
model.compile(loss='mean_squared_error', optimizer = 'adam')
model.summary()       # Print out architecture.
```

### Training the GRU predictive model

```
history = model.fit(X[:], Y[:],
                    epochs=50,
                    batch_size=32,
                    validation_split=0.2,
                    verbose=1) # Verbose outputs data
```

## Model validation

```
plt.plot(history.history['loss'], label = 'training loss')
plt.plot(history.history['val_loss'], label ='validation loss')
plt.legend()
```

## To save the trained model

```
model.save(GRU_pred_Model_ADPCM.h5')
```

## Preparing and loading testing set

```
testX, testY =  X[:], Y[:]
```

## GRU model one step-based prediction

```
plt.figure(figsize=(12, 4))
y_pred = loaded_model.predict(testX)

# Oreginal speech signal data
plt.title('The Original PCM and predicted Speech Signal')
plt.plot(testY, 'b', label='Original Signal sample')

# Predicted speech signal data
plt.xlabel('Number of samples')
plt.ylabel('Amplitude (Quantization level)')
plt.plot(y_pred, 'g', label='predicted Speech Signal')
plt.legend(loc='upper center')
```

## Evalout the model using SNR

```
GRU_err = testY[:] - y_pred[:]

plt.figure(figsize=(12, 4))
plt.plot(GRU_err, label = 'error_signal')
plt.legend()

GRU_mse = 10*np.log10(np.mean(pow(np.abs(GRU_err[:]),2)))
GRU_sigpow = 10*np.log10(np.mean(pow(np.abs(testY),2)))

#print(dnn_mse, dnn_sigpow, lms_mse, lms_sigpow)
print("GRU predictor SNR:", GRU_sigpow - GRU_mse, "dB")
```

### Abbreviations

| | |
|---|---|
| ADPCM | Adaptive differential pulse code modulation |
| dB | Decibel |
| CELP | Code-excited linear prediction |
| DARPA | Defense advanced research projects agency |
| DPCM | Differential pulse code modulation |
| GRU | Gated recurrent unit |
| LPC | Linear predictive coding |
| MOS | Mean opinion score |
| MSE | Mean squared error |
| IMA | Interactive multimedia association |
| SNR | Signal-to-noise ratio |
| SEGSNR | Segmental signal-to-noise ratio |
| PCM | Pulse code modulation |
| POLQA | Perceptual objective listening quality assessment |
| TIMIT | Texas instruments misspoken telephone corpus |

### Acknowledgements
I would like to express my gratitude to the German Academic Exchange Service (DAAD) for providing funding for this study, including support for tuition fees, research expenses, and a stipend for living expenses. I am also grateful to Jomo Kenyatta University of Agriculture and Technology (JKUAT) for hosting me to study and for providing invaluable academic resources and support.

### Authors' contributions
The author contributions for this manuscript are as follows: conceptualization was performed by G.K. and W.M.; methodology was developed by G.K.,W.M, M.K, and A.M.; experiment and testing was conducted by G.K.; writing of the original draft was prepared by G.K.; writing a review and editing were performed by W.M., M.K., and A.M.; supervision was provided by W.M., M.K., and A.M. All authors have read and approved the final version of the manuscript.

### Funding
This research was supported by a DAAD scholarship, which provided funding for tuition fees, research expenses, and a stipend for the author during my Ph.D. studies at Jomo Kenyatta University of Agriculture and Technology (JKUAT), Kenya.

### Availability of data and materials
The dataset was used in this study for training and testing the model is available in the DARPA TIMIT continuous speech acoustic-phonetic corpus [42]: https://www.kaggle.com/datasets/mfekadu/darpa-timit-acousticphonetic-continuous-speech.

## Declarations

### Competing interests
The authors declare that they have no competing interests.

Received: 23 March 2023 Accepted: 12 December 2023
Published online: 20 January 2024

### References
1. S.K. Jagtap, M.S. Mulye, M.D. Uplane, Speech coding techniques. Procedia Computer Science. **49**, 253–263 (2015)
2. M. Cernak, A. Asaei, A. Hyafil, Cognitive speech coding: examining the impact of cognitive speech processing on speech compression. IEEE Signal Process. Mag. **35**(3), 97–109 (2018)
3. K. Zhen, J. Sung, M.S. Lee, S. Beack, M. Kim, Scalable and efficient neural speech coding: a hybrid design. IEEE/ACM Trans. Audio Speech Lang. Process. **30**, 12–25 (2021)
4. A. Mustafa, J. Buthe, S. Korse, K. Gupta, G. Fuchs, N. Pia, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. A streamwise gan vocoder for wideband speech coding at very low bit rate (2021), pp. 66–70. https://doi.org/10.1109/WASPAA52581.2021.9632750
5. S. Mishra, in *Proceedings of the International Conference on Recent Trends in Information Technology and Computer Science*. A survey paper on different data compression techniques (2016), pp. 738–740
6. W.B. Kleijn, F.S. Lim, A. Luebs, J. Skoglund, F. Stimberg, Q. Wang, T.C. Walters, in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*. WaveNet-based low-rate speech coding (2018), pp. 676–680. https://doi.org/10.1109/ICASSP.2018.8462529
7. A. Wahyudi, A. Omondi, in *Proceedings Euromicro Symposium on Digital System Design: Architectures, Methods and Tools*. Parallel multimedia processor using customised infineon tricores (IEEE, Dortmund, 2002), pp. 140–147. https://doi.org/10.1109/DSD.2002.1115362
8. B. Hatipoğlu. A wireless entryphone system implementation with MSP430 and CC1100 (2008)
9. A. Linley, M. Field, A. Bruce, ADPCM with a PIC32. Technology **10**(1), 1–18 (2017)

10. A.L. Todorovic, in *Telev. Technol. Demystified*. Digital audio compression methods (2020), pp. 102–115. https://doi.org/10.4324/9780080457062-11
11. R. Richey, Adaptive differential pulse code modulation using PICmicroTM microcontrollers. Technology **10**(1), 1–41 (1997). https://doi.org/10.9790/4200-10011519
12. Z. Zhao, H. Liu, T. Fingscheidt, in *Proc. Eur. Signal Process. Conf. (EUSIPCO)*. Nonlinear prediction of speech by echo state networks (2018), pp. 2085–2089. https://doi.org/10.23919/EUSIPCO.2018.8553190
13. H. Zhao, J. Zhang, Pipelined Chebyshev functional link artificial recurrent neural network for nonlinear adaptive filter. IEEE Trans. Syst. Man Cybern. Part B Cybern. **40**(1), 162–172 (2010). https://doi.org/10.1109/TSMCB.2009.2024313
14. T. Yoshimura, K. Hashimoto, K. Oura, Y. Nankaku, K. Tokuda, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Speaker-dependent WaveNet-based delay-free ADPCM speech coding (2019), pp. 7145–7149
15. C. Garbacea, et al., in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2019-May. Low bit-rate speech coding with VQ-VAE and a WaveNet decoder (Brighton, UK, 2019), pp. 735–739. https://doi.org/10.1109/ICASSP.2019.8683277
16. S. Amada, et al., Experimental evaluation of WaveRNN predictor for audio lossless coding (2019), pp. 315–318. http://www.tara.tsukuba.ac.jp/~maki/reprint/Makino/amada19ncsp315-318.pdf
17. J.M. Valin, J. Skoglund, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2019-May. LPCNet: improving neural speech synthesis through linear prediction (2019), pp. 5891–5895. https://doi.org/10.1109/ICASSP.2019.8682804
18. R. Prenger, R. Valle, B. Catanzaro, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2019-May. Waveglow: a flow-based generative network for speech synthesis (2019), pp. 3617–3621. https://doi.org/10.1109/ICASSP.2019.8683143
19. R. Yamamoto, E. Song, J.M. Kim, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2020-May. Parallel WaveGAN: a fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram (2020), pp. 6199–6203. https://doi.org/10.1109/ICASSP40776.2020.9053795
20. H. Ze, A. Senior, M. Schuster, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Statistical parametric speech synthesis using deep neural networks (2013), pp. 7962–7966. https://doi.org/10.1109/ICASSP.2013.6639215
21. S. Latif, R. Rana, J. Qadir, J. Epps, Variational autoencoders for learning latent representations of speech emotion: a preliminary study (2017)
22. M. Blaauw, J. Bonada, in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, vol. 08-12-Sept. Modeling and transforming speech using variational autoencoders (2016), pp. 1770–1774. https://doi.org/10.21437/Interspeech.2016-1183
23. A. Mustafa, N. Pia, G. Fuchs, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2021-June. Audio codec enhancement with generative adversarial networks (2021), pp. 6034–6038. https://doi.org/10.1109/ICASSP39728.2021.9413605
24. T. Kaneko, S. Takaki, H. Kameoka, J. Yamagishi, in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, vol. 2017-Augus, Generative adversarial network-based postfilter for STFT spectrograms (2017), pp. 3389–3393. https://doi.org/10.21437/Interspeech.2017-962
25. A. Défossez, J. Copet, G. Synnaeve, Y. Adi, High fidelity neural audio compression (2022)
26. N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, M. Tagliasacchi, Soundstream: An end-to-end neural audio codec. IEEE/ACM. Audio Speech Lang. Process. **30**, 495–507 (2022). https://doi.org/10.1109/TASLP.2021.3129994
27. M. Faúndez-Zanuy, in *Lect. Notes Comput. Sci.*, vol. 2687. Non-linear speech coding with MLP, RBF and Elman based prediction (2003), pp. 671–678. https://doi.org/10.1007/3-540-44869-1_85
28. M. Faúndez-Zanuy, in *Proc. Int. Workshop on Applications of Neural Networks to Telecommunications*, Nonlinear vectorial prediction with neural nets 1 (2001), pp. 754–761. https://doi.org/10.5555/646370.688874
29. A. Wang, Z. Sun, X. Zhang, in *Proc. World Congr. Intell. Control Autom.* A non-linear prediction speech coding system based on ANN (2002), pp. 607–611. https://doi.org/10.1109/wcica.2002.1022183
30. S.H.L. Li, in *Proc. Int. Workshop on Applications of Neural Networks to Telecommunications*. Kbps adaptive differential pulse code modulation of speech (2013), pp. 142–148
31. M. Faúndez-Zanuy, F. Vallverdu, E. Monte, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1. Nonlinear prediction with neural nets in ADPCM (1998), pp. 345–348. https://doi.org/10.1109/ICASSP.1998.674438
32. S.C.H. Hoi, D. Sahoo, J. Lu, P. Zhao, Online learning: a comprehensive survey. Neurocomputing **459**, 249–289 (2021). https://doi.org/10.1016/j.neucom.2021.04.112
33. R.J.'Kuo, B. Prasetyo, B.S. Wibowo, in *Proceedings of the IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA)*. Deep learning-based approach for air quality forecasting by using recurrent neural network with Gaussian process in Taiwan (2019), pp. 471–474. https://doi.org/10.1109/IEA.2019.8715113
34. R. Lotfidereshgi, P. Gournay, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Speech prediction using an adaptive recurrent neural network with application to packet loss concealment (2018), pp. 5394–5398
35. S. Nosouhian, F. Nosouhian, A.K. Khoshouei, A review of recurrent neural network architecture for sequence learning: comparison between LSTM and GRU. Preprints **202107.0252**, 1–7 (2021). https://doi.org/10.20944/preprints202107.0252.v1
36. A. Alqushaibi, S.J. Abdulkadir, H.M. Rais, Q. Al-Tashi, in *Proceedings of the International Conference on Computational Intelligence (ICCI)*. A review of weight optimization techniques in recurrent neural networks (2020), pp. 196–201. https://doi.org/10.1109/ICCI51257.2020.9247757
37. L. Qu, J. Lyu, W. Li, D. Ma, H. Fan, Features injected recurrent neural networks for short-term traffic speed prediction. Neurocomputing **451**, 290–304 (2021). https://doi.org/10.1016/j.neucom.2021.03.054
38. A. Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Phys. D Nonlinear Phenom. **404**, 132306 (2020). https://doi.org/10.1016/j.physd.2019.132306
39. W. Yan, J. Zhang, S. Zhang, P. Wen, A novel pipelined neural IIR adaptive filter for speech prediction. Appl. Acoust. **141**, 64–70 (2018). https://doi.org/10.1016/j.apacoust.2018.06.007
40. Z.H. Ling, Y. Ai, Y. Gu, L.R. Dai, Waveform modeling and generation using hierarchical recurrent neural networks for speech bandwidth extension. IEEE/ACM Trans. Audio Speech Lang. Process. **26**(5), 883–894 (2018). https://doi.org/10.1109/TASLP.2018.2798811
41. S. Mehri, et al., in *Proceedings of the 5th International Conference on Learning Representations*. SampleRNN: an unconditional end-to-end neural audio generation model (2017), pp. 1–11
42. A. Pfalz, Generating audio using recurrent neural networks LSU Digit. Commons. (2018)
43. N. Chanchaochai, et al., GlobalTIMIT: acoustic-phonetic datasets for the world's languages. (Linguistic Data Consortium, University of Pennsylvania, USA, Unknown)
44. L.F. Lamel, W.M. Fisher, J.G. Fiscus, D.S. Pallett, N.L. Dahlgren, DARPA TIMIT. Unknown (1990)
45. S. Uhrig, in *T-Labs Series in Telecommunication Services*. Speech quality assessment (2022), pp. 21–46. https://doi.org/10.1007/978-3-030-71389-8_3
46. P. Papadopoulos, A. Tsiartas, J. Gibson, S. Narayanan, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, A supervised signal-to-noise ratio estimation of speech signals (2014), pp. 8237–8241. https://doi.org/10.1109/ICASSP.2014.6855207


## Publisher's Note