A Novel Hybrid Architecture Combining CNN, LSTM, and Attention Mechanisms for Automatic Speech Recognition

Wenye Song
wenye.song@emory.edu
Emory University
Decatur, GA, USA

Abstract

This research proposes a robust sequence-to-sequence model architecture for speech recognition tasks, incorporating convolutional and recurrent neural networks (CNN and RNN) with attention mechanisms. The encoder utilizes a hybrid approach of residual convolutional blocks (RCNN) and bidirectional Long Short-Term Memory (BLSTM) networks to extract hierarchical temporal and spectral features from input sequences efficiently. The decoder employs an attention-augmented LSTM with Luong Attention, dynamically focusing on relevant parts of the encoded representation during sequence generation. To enhance generalization and robustness, during data preprocessing, data augmentation techniques such as phase randomization and time-frequency masking are integrated. Features are further refined using Mel-Frequency Cepstral Coefficients (MFCC) and Filter Banks (FBANK) combined with delta and delta-delta coefficients. The model is trained on the TIMIT dataset and achieves a Phoneme Error Rate (PER) of 16.7%, demonstrating the effectiveness of the proposed architecture and preprocessing strategies in capturing complex phoneme patterns and improving recognition performance.

Keywords

Automatic Speech Recognition, Audio Signal Process, Deep Learning, CNN, BLSTM, Attention Mechanism

1 Introduction

Automatic Speech Recognition (ASR) is a technology that transcribes spoken language into text. It has broad applications ranging from voice assistants and transcription services to real-time language translation and accessibility tools. The goal of ASR systems is to accurately transcribe audio input while accommodating variability in accents, background noise, and speaking styles.

Over the years, ASR has evolved through several technological advancements. Early speech recognition technology relied on Hidden Markov Models (HMMs) combined with Gaussian Mixture Models (GMMs) [1]. However, these approaches struggled with complex patterns in speech data. Deep learning has been applied in ASR, with Deep Neural Networks (DNNs) providing more robust feature extraction and representation [2].

Convolutional Neural Networks (CNNs), with their ability to capture local spatial correlations in spectrogram representations of speech signals, have achieved significant success in overcoming the challenges posed by background noise and speaker variability [3].To be specific, human speech signals have local correlations along the time axis (short segments of speech) and the frequency axis (between frequency components). CNN can capture those local dependencies across both time and frequency dimensions.

Further advancements included Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks [5], which improved the ability to model long-term dependencies in speech. Unlike DNNs and CNNs, which primarily process data with fixed spatial relationships, RNNs and LSTMs are specifically designed to handle sequential data by maintaining a hidden state that evolves over time. This enables them to retain contextual information from earlier sequences and use it to influence the processing of later sequences.

Attention mechanism has been also used in ASR field. Several works designed Attention-based Encoder-Decoder architecture for ASR tasks to improve the model's ability to catch the relevant information from the whole audio sequence when predict each audio token [10] [12] [18] [19]. Recently, Transformer-based models [10] leveraging self-attention mechanisms have emerged as state-of-the-art techniques, offering superior parallelization and the ability to capture global dependencies in both audio and text sequences.

In recent advancements in ASR, integrating large language models (LLMs) has shown significant promise. Dong et al. [16] proposed Seed-ASR, which leverages contextual information encoded through LLMs to improve ASR performance across diverse datasets and scenarios, achieving substantial reductions in error rates compared to traditional end-to-end models. Similarly, Yu et al. [17] explored novel connectors between speech encoders and LLMs, demonstrating consistent improvements in word error rate (WER) reduction, particularly with their Q-Former structure.

In addition to these advancements, Yu Zhang et al. introduced the Universal Speech Model (USM), a large-scale ASR system pretrained on a multilingual dataset spanning over 100 languages and 12 million hours of audio. This innovation highlights the potential of multilingual and modality-matching techniques in scaling ASR systems for diverse languages and scenarios[20].

As ASR technology continues to advance, the focus remains on improving recognition accuracy, computational efficiency, and robustness across diverse acoustic environments. To evaluate the performance of ASR systems, metrics such as Word Error Rate (WER) and Phoneme Error Rate (PER) are widely used. PER measures the proportion of phoneme-level errors (insertions, deletions, and substitutions) relative to the total number of phonemes in the reference transcription. A lower PER indicates a more accurate system and is often preferred for tasks requiring fine-grained phonetic analysis. In contrast, the WER measures the word-level errors. The key difference between WER and PER is that WER measures errors

at the word level, while PER evaluates them at the phoneme level, making WER more relevant for semantic accuracy and PER more granular for pronunciation accuracy.

2 Related Work

2.1 CNN for ASR

CNNs have been successfully applied to ASR by extracting robust features from speech data. Rather than employing fully connected hidden layers like HMM and DNN, CNNs introduce a structure made up of alternating convolutional and pooling layers. Abdel-Hamid et al. introduced a CNN-based ASR model focusing on the local temporal and spectral features of the input. CNNs significantly reduce error rates compared to DNNs, demonstrating their outstanding performance in extracting information from speech signals [3]. However, traditional CNNs in ASR often use fewer than 10 layers, limiting their ability to capture complex speech features. Wang et al. proposed an architecture called RCNN-CTC, introducing a deeper CNN with residual connections, improving learning efficiency. It employs the Connectionist Temporal Classification (CTC) loss function for end-to-end training tailored to sequential tasks. A CTC-based system combination further reduces WER, achieving strong results on the WSJ and Tencent Chat datasets [4].

2.2 RNN, LSTM, and Hybrid Models for ASR

Recurrent Neural Networks (RNNs) are a type of neural network architecture designed to process sequential data by sharing information across time steps. Beyond CNNs, RNNs are another option for addressing ASR tasks, which involve time series. Graves et al. introduced deep RNNs, particularly Long Short-Term Memory (LSTMs), to capture long-range contexts in speech data [7]. LSTM introduces memory cells and gating mechanisms, allowing it to overcome the vanishing gradient problem and capture long-term dependencies more effectively [5].

The RNN-Transducer model proposed by Rao et al. combines acoustic and language models, achieving high WER accuracy by processing sequences directly from audio [6]. Similar to RNN-T, Zhang et al. proposed a Transformer Transducer model for end-to-end speech recognition, combining Transformer encoders with RNN-T loss and demonstrating its effectiveness for streaming recognition on the LibriSpeech dataset [8]. Recently, hybrid architectures combining CNNs and Bidirectional LSTM (BLSTM) have emerged. For example, CNN-BLSTM models combine CNNs' feature extraction capabilities with BLSTM's ability to handle temporal dependencies [9].

2.3 Attention-based Encoder-Decoder Model for ASR

The attention-based Encoder-Decoder model has become one of the most prominent sequence-to-sequence architectures for ASR tasks, which consists of an encoder, decoder, and attention mechanism. An encoder-decoder model in ASR is a neural network architecture where the encoder processes the input audio features into a compact representation, and the decoder converts this representation into the corresponding textual transcription. The attention mechanism assigns scores to the audio information representation generated by the encoder. These scores represent the relationship between each encoded audio feature and the candidate output unit at the current decoding step. Using these scores, the decoder dynamically focuses on the most relevant audio features, enhancing its ability to generate accurate predictions [10]. Xue et al. worked on ASR with an Attention-based Encoder-Decoder architecture by adding location constraint vectors and developing location-constrained attention mechanisms, achieving WER reductions of 10.6% and 16.7% [13]. Qijia Li et al. developed a hybrid system combining DNN-HMM-based ASR and attention-based encoder-decoder models through lattice rescoring. This two-pass framework effectively improves WER by leveraging the strengths of both approaches, achieving competitive performance on standard ASR benchmarks [18]. Xun Gong et al. introduced the Factorized Attention-Based Encoder-Decoder model for domain-adaptive ASR, addressing challenges in text-only domain adaptation. By integrating a domain-specific language model with Attention-Based Encoder-Decoder model, their approach achieves significant WER reductions in both in-domain and out-of-domain scenarios [19].

2.4 Works based on the TIMIT Dataset

In this research, the TIMIT Acoustic-Phonetic Continuous Speech Corpus dataset [15] is used for training the model. Several previous works have been done on this dataset. Graves et al. introduced an end-to-end probabilistic sequence transduction model using RNNs, addressing the challenge of learning alignments between input and output sequences without pre-defining them, validated on the TIMIT dataset, achieving a PER of 23.8% [14]. Graves et al. later used deep RNNs with CTC and LSTM architectures to achieve a 17.7% PER on the TIMIT dataset [7]. Toledano DT et al. using DNN on TIMIT dataset had a result of PER = 18.25% [11]. Chorowski et al. used an attention mechanism with location awareness, achieving a PER of 17.6% [12]. This research proposes an approach combining convolutional layers and recurrent networks with attention to further improve phoneme recognition performance on TIMIT, aiming for competitive results against state-of-the-art methods.

3 Methodology

3.1 Data Preparation

3.1.1 Dataset Introduction

TIMIT Acoustic-Phonetic Continuous Speech Corpus dataset is a well-established dataset widely used for acoustic-phonetic studies and the development and evaluation of automatic speech recognition (ASR) systems. It consists of recordings from 630 speakers, and each speaker read ten phonetically diverse sentences, including eight major dialect regions of American English. The corpus also provides detailed speaker metadata, including attributes such as gender, dialect region, birth date, height, race, and education level. The speaker distribution is approximately 70% male and 30%

female. The rich phonetic diversity, combined with comprehensive demographic metadata, makes TIMIT an ideal dataset for ASR tasks [15].

The TIMIT dataset preprocessing pipeline is designed to prepare audio data for training, validation, and testing in a sequence-to-sequence model. The process involves several steps, including data augmentation, feature extraction, phoneme label alignment, and data normalization. The processed data is serialized into `TFRecord` format to ensure efficient access during training. The preprocessing pipeline is described below in detail.

3.1.2 Short-Time Fourier Transform

The raw audio signal in TIMIT is in the form of `.wav`, containing the digitized time-domain representation $x[n]$ for each speech sample, where n is the index of the sampling point in the audio signal.

Here, $x[n]$ is transformed into the frequency domain using the Short-Time Fourier Transform (STFT), enabling clearer extraction of frequency features, which are critical for machine learning models to recognize and classify sound patterns:

$$X(k,t) = \sum_{n=0}^{N-1} x[n]w[n-tH]e^{-j2\pi kn/N},$$

where n is the index for each audio signal, k is the frequency bin, t is the time frame index, $w[n]$ is the Hann window, H is the hop size, N is the FFT size, and j is the imaginary unit.

Parameter	Value	Meaning	Explanation
FFT Size (N)	400	Frequency resolution	Window size of 25 ms
Hop Size (H)	160	Sliding by 160 points	Provides time resolution to capture dynamics
Window Size (L)	400	Each window covers 400 samples	Matches FFT size and spans a single phoneme
Window Function ($w[n]$)	Hann	Smooths frame edges	Commonly used in speech signal processing

Table 1: STFT Parameters in this task

3.1.3 Data Augmentation

To enhance model robustness and reduce overfitting, data augmentation is applied to the audio signals and is only conducted over the training dataset. This involves two main techniques: phase randomization and masking.

3.1.3.1 Phase Randomization The phase $\phi(k,t)$ of the STFT is then perturbed by a random scaling factor $\mu \sim \mathcal{N}(1,\delta)$:

$$\phi'(k,t) = \mu \cdot \phi(k,t), \quad \mu \sim \mathcal{N}(1,\delta).$$

The signal is reconstructed using the inverse STFT:

$$x'[n] = \text{ISTFT}(|X(k,t)|e^{j\phi'(k,t)}).$$

3.1.3.2 Frequency and Time Masking Random regions in the frequency and time domains are set to zero.

- Frequency Masking: A band of random width w is zeroed out, where $w \sim \text{Uniform}(10,30)$.
- Time Masking: A random time segment of width $w \sim \text{Uniform}(5, T/9)$ is zeroed out, where T is the total number of time frames.

These augmentations improve the model's generalization by simulating variability in real-world audio signals.

3.1.4 Feature Extraction through MFCC and FBANK

Two types of features are extracted from the audio signals both for the training set and test set: Mel-Frequency Cepstral Coefficients (MFCC) and Filter Banks (FBANK). Both methods capture spectral and temporal information from the audio.

MFCC Extraction The MFCC features are computed as follows:

(1) The signal is transformed into the Mel-spectral domain $M(f,t)$ using Mel filter banks. The time-domain signal $x'[n]$ is transformed into the frequency-domain signal $X(k,t)$ via Short-Time Fourier Transform (STFT).
(2) The log of the filter bank energies is taken:

$$\log M(f,t) = \log\left(\sum_{k=1}^{K} |X(k,t)|^2 H_k(f)\right),$$

where $H_k(f)$ is the Mel filter response for the k-th frequency bin at the Mel frequency f.
(3) A Discrete Cosine Transform (DCT) is applied to decorrelate the coefficients:

$$C_t[f] = \sum_{i=1}^{F} \log M(f,t) \cos\left(\frac{\pi(2f-1)}{2F}\right).$$

where F is the number of Mel filterbanks used in the Mel-frequency analysis.

Delta and Delta-Delta Features Temporal derivatives of MFCCs are calculated to capture dynamic changes over time. The Δ-MFCC and ΔΔ-MFCC are defined as:

$$\Delta C_t = \frac{\sum_{n=1}^{N} n(C_{t+n} - C_{t-n})}{2\sum_{n=1}^{N} n^2}, \quad \Delta\Delta C_t = \Delta(\Delta C_t).$$

where C_t is the MFCC vector at time t.

FBANK Features Similarly, the filter bank energies are logged and augmented with their delta and delta-delta coefficients to form a comprehensive feature vector.

3.1.5 Phoneme Label Alignment

Each audio file is accompanied by a `.phn` file, which contains phoneme annotations. These annotations are aligned with the audio signal and mapped to a predefined phoneme index set P. The phoneme sequence for each utterance is parsed as:

$$P = \{p_1, p_2, \ldots, p_N\}, \quad p_i \in P_{61},$$

where P_{61} represents the 61-phoneme set in the TIMIT dataset that covers all possible phonemes in American English, which is :

phn61 = ['aa', 'ae', 'ah', 'ao', 'aw', 'ax', 'ax-h', 'axr', 'ay', 'b', 'bcl', 'ch', 'd', 'dcl', 'dh', 'dx', 'eh', 'el', 'em', 'en', 'eng', 'epi', 'er', 'ey', 'f', 'g', 'gcl', 'h#', 'hh', 'hv', 'ih', 'ix', 'iy', 'jh', 'k', 'kcl', 'l', 'm', 'n', 'ng', 'nx', 'ow', 'oy', 'p', 'pau', 'pcl', 'q', 'r', 's', 'sh', 't', 'tcl', 'th', 'uh', 'uw', 'ux', 'v', 'w', 'y', 'z', 'zh']

3.1.6 Normalization

To ensure consistent feature scaling, the extracted features are normalized using a standard scaler:

$$z = \frac{x - \mu}{\sigma},$$

where μ and σ are the mean and standard deviation computed over the features matrix M of the training dataset.

3.2 Model Architecture

The model adopts a sequence-to-sequence (seq2seq) framework with an encoder-decoder structure. A seq2seq model is designed to transform an input sequence into an output sequence. In this task, the input is a sequence of an audio feature matrix, and the output is a sequence of transcribed texts tokens. The encoder in this model extracts features from the input by processing the sequence into a compact context vector, while the decoder generates the output transcription by converting the context vector into a sequence of target phenomes.

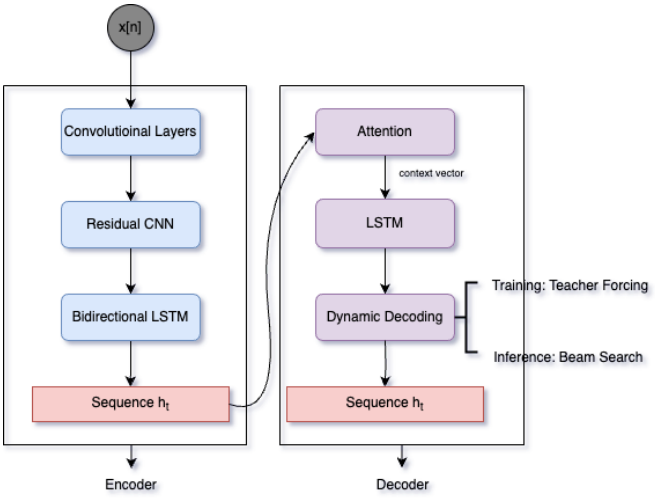

Figure 1: Architecture Diagram

3.2.1 Encoder

The encoder extracts meaningful features from input audio sequences and encodes their temporal dependencies. It consists of convolutional layers followed by bidirectional long-short-term memory (LSTM) layers.

3.2.1.1 Convolutional Feature Extraction The input feature matrix M has a size of $T \times F$, where T is the number of time frames and F is the number of feature dimensions. Through the convolution process, M is split into three equal-sized feature channels:

$$M' = \text{Stack}(M_1, M_2, M_3).$$

A stack of convolutional blocks captures local patterns in the input. Each convolution block applies a 2D convolution followed by batch normalization, ReLU activation, and dropout. The equation for convolution with dropout and batch normalization is as follows:

$$H_l = \text{Dropout}(\text{ReLU}(\text{BatchNorm}(W_l * H_{l-1})))$$

where $*$ denotes the convolution operator, H_{l-1} is the input equal to M, H_l is the output, and W_l is a 3×3 convolutional kernel containing learnable weights of the l-th layer. The convolutional stride is set to $(1, 3)$. With the convolution operation, the time dimension T is reduced to $T/3$.

3.2.1.2 Residual Convolution The residual convolution leverages residual connections, allowing the network to maintain gradient flow even in deeper layers. The residual block is defined as:

$$H_l = H_{l-1} + F(H_{l-1}, W_l)$$

where $F(H_{l-1}, W_l)$ represents the stacked nonlinear transformations and W_l are the corresponding weights.

By adding the input H_{l-1} to the output of $F(H_{l-1}, W_l)$, gradients can flow freely during backpropagation. The advantage is mathematically evident from:

$$\frac{\partial L}{\partial H_{l-1}} = \frac{\partial L}{\partial H_l}\left(1 + \frac{\partial F}{\partial H_{l-1}}\right)$$

where the term 1 ensures that gradients do not vanish even if $\partial F / \partial H_{l-1}$ becomes small.

3.2.1.3 Bidirectional LSTM The output of the residual connection H_l is transformed into a suitable input for the bidirectional LSTM (BLSTM). BLSTM captures temporal dependencies in both forward and backward directions. The transformation to BLSTM input X is:

$$X_t = \text{BLSTM}(H_t)$$

The BLSTM consists of:

Forget Gate: Decides how much of the previous state to forget.

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f)$$

where σ is the sigmoid function, x_t is the current input, h_{t-1} is the hidden state from the previous time step, W_f and b_f are the weight and bias for the forget gate.

Input Gate: Determines how much new information from the current input x_t should be added to the cell state:

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i)$$
$$\tilde{C}_t = \tanh(W_c[x_t, h_{t-1}] + b_c),$$

where $\tilde{C}_t$ is the candidate cell state.

Cell State Update: Combines the forget gate and input gate to update the cell state:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

where $\odot$ denotes element-wise multiplication.

Output Gate: Determines how much of the cell state should influence the hidden state h_t:

$$o_t = \sigma(W_o[x_t, h_{t-1}] + b_o)$$
$$h_t = o_t \odot \tanh(C_t).$$

The final output of the encoder is h_t, a sequence with size of $batch\_size \times T/3 \times 2 \cdot num\_encoder$, where $T/3$ represents the reduced time steps due to convolutional down-sampling, and the factor of 2 arises from the bidirectional nature of the LSTM. The BLSTM processes the input sequence in both forward and backward directions, allowing the model to capture dependencies across the entire sequence contextually.

3.2.2 Decoder: Dynamic Decoding with Attention

3.2.2.1 Attention Mechanism The Luong attention mechanism computes alignment scores between the decoder's current hidden state h_t and each encoder output H_s:

$$e_{t,s} = h_t^\top W_a H_s$$

where σ is the sigmoid function, x_t is the current input, h_{t-1} is the hidden state from the previous time step, W_f and b_f are the weight and bias for the forget gate.

The attention weights are calculated as:

$$\alpha_{t,s} = \frac{\exp(e_{t,s})}{\sum_{s'} \exp(e_{t,s'})}$$

The context vector is:

$$c_t = \sum_s \alpha_{t,s} H_s$$

3.2.2.2 LSTM Decoder The context vector c_t integrates into the decoder LSTM:

$$h_t = \text{LSTM}([e_{t-1}, c_t, h_{t-1}])$$

where c_t is the context vector from the attention mechanism, e_{t-1} is the embedding of ground truth of target sequence in the training process and previous predicted token in the inference step.

3.2.2.3 Dynamic Decoding During training, the decoder uses teacher forcing with ground-truth tokens:

$$P(y_t \mid y_{<t}, x) = \text{Softmax}(W_o h_t)$$

where $W_o \in \mathbb{R}^{p \times d}$ is a learnable weight matrix, p being the phoneme size, and d the dimensionality of the hidden state. h_t is the decoder's hidden state at time step t, y_t is the predicted output token at time step t, given the previous ground truth tokens $y_{<t}$ and encoded sequence x. By the softmax function, the logits $W_o h_t$ are converted to a probability distribution over the candidate phonemes.

During inference, y_t is fed back into the decoder as the next step's input. Beam search is used here to generate the most likely sequence by maintaining k candidate sequences and selecting the best based on cumulative probabilities.

3.2.3 Loss Function and Optimization

The model minimizes cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T_i} w_{i,t} \log P(y_{i,t} | y_{<t}, x_i)$$

where N is the size of each batch, i is index of each sample in a batch, t is the index of time, w is the weight based on each sample I and time t, y is the ground truth of each sample i at time t, and x is the input sequence of sample i.

4 Experiment & Analysis

4.1 Hyperparameters of Training

The training process is configured with the following parameters:

Learning Rate: Set to 0.001.

Optimizer: Adam.

Epochs: The model is trained for 40 epochs.

4.2 Training Process

The experiment was performed on the TIMIT dataset. Each epoch began by shuffling the training dataset with a unique seed to ensure diverse batch sampling, and the training graph performed forward and backward passes to update model parameters by minimizing the cross-entropy loss. Checkpoints were saved at the end of each epoch, enabling model evaluation and fine-tuning at later stages. During each epoch, the training loss was computed as the average over all batches, and the model's performance was evaluated on a separate development set. The evaluation graph restored the latest checkpoint and computed the development loss over the entire dataset without applying gradient updates or dropout. Metrics such as training loss, development loss, and epoch time were logged to monitor model performance and generalization. This process helped track training progress and ensured the training was consistent and efficient.

4.3 Results & Analysis

For results in ASR, Phoneme Error Rate (PER) is often used to evaluate the performance of speech recognition models:

$$\text{PER} = \frac{S + D + I}{N} \tag{1}$$

where:

- S: Substitutions — the number of phonemes in the prediction that do not match the target phonemes.
- D: Deletions — the number of target phonemes that do not appear in the prediction.
- I: Insertions — the number of extra phonemes in the prediction.
- N: Number of Reference Phonemes — the total length of the target phoneme sequence.

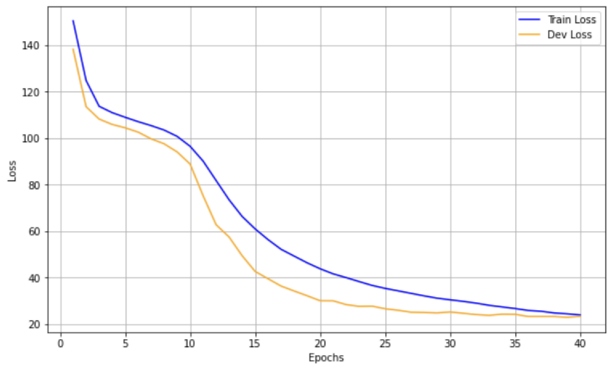

Figure 2: Training and validation loss curves across epochs.

Figure 2 shows the reduction in training and development loss during the model's training process, demonstrating convergence and generalization.

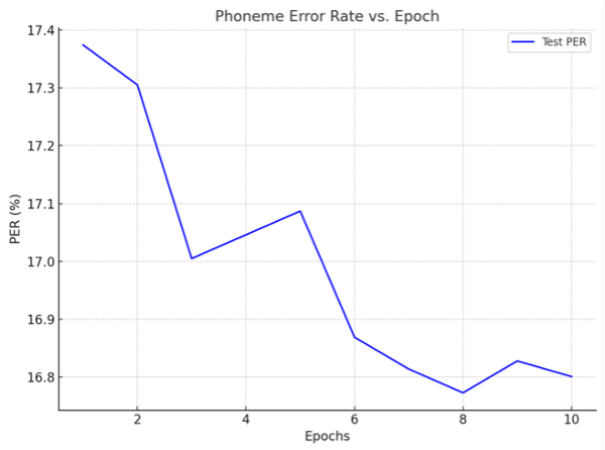

Figure 3: Phoneme Error Rate (PER) on the test set across epochs.

Figure 3 shows the model's performance improvement over the first 10 epochs in terms of phoneme recognition accuracy.

The training and validation loss curves (Figure 2) show a consistent decline as training progresses, indicating effective learning. The gap between training and validation loss remains small, suggesting that the model generalizes well without overfitting. Meanwhile, the Phoneme Error Rate (PER) on the test set (Figure 3) exhibits a clear downward trend during the first 10 epochs, reflecting improved phoneme recognition performance as the model learns to align input features with target transcriptions. The best PER achieved 16.7%. These results demonstrate the effectiveness of the training pipeline and the model's ability to accurately transcribe phonemes.

5 Conclusion

This architecture demonstrates a robust ability to handle complex sequence-to-sequence tasks by combining efficient feature extraction, contextual encoding, and dynamic decoding. The interplay between CNNs, residual blocks, BLSTMs, and attention mechanisms results in improved accuracy and generalization. However, further optimization may be required for scalability to larger datasets and real-time inference scenarios.

5.1 Limitations

There are still several areas the model can be improved. The architecture relying on recurrent layers, such as BLSTMs, increases training time compared to more modern architectures like Transformers. Beam search decoding and attention mechanisms require significant memory, making the model less efficient for long sequences or large batch sizes. Also, the model requires extensive labeled data to achieve high performance, which may not always be feasible in low-resource languages or domains.

5.2 Future Steps

While this model is built on recurrent layers, Transformer models have shown improvements in sequence-to-sequence tasks due to their parallelism and ability to handle long-range dependencies. Future work could involve replacing BLSTMs with Transformer encoders. In addition, implementing memory-efficient beam search strategies is considered, such as pruning less likely candidates or limiting the beam width dynamically based on context, can help reduce the memory overhead associated with attention mechanisms and decoding. To alleviate the dependency on extensive labeled datasets, more data augmentation techniques, such as synthetic data generation, can be explored in this task. Furthermore, extending the model to handle multimodal inputs could unlock more powerful applications, such as combining audio with text or video and conducting emotion recognition in conversations.

6 References

References

[1] D. Su, X. Wu, and L. Xu, "GMM-HMM acoustic model training by a two-level procedure with Gaussian components determined by automatic model selection,"

IEEE ICASSP, 2010.
[2] Zhi-Jie Yan, Qiang Huo, and Jian Xu, "A scalable approach to using DNN-derived features in GMM-HMM based acoustic modeling for LVCSR," *Interspeech*, 2013.
[3] O. Abdel-Hamid et al., "CNN for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014.
[4] Y. Wang et al., "Residual convolutional CTC networks for automatic speech recognition," Tencent AI Lab, 2017.
[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, 1997.
[6] K. Rao et al., "Exploring architectures for streaming ASR with RNN-Transducer," *IEEE ICASSP*, 2018.
[7] A. Graves et al., "Speech recognition with deep recurrent neural networks," *ICML*, 2013.
[8] Q. Zhang et al., "Transformer transducer: A streamable speech recognition model," *IEEE ICASSP*, 2020.
[9] M. Dar and J. Pushparaj, "Hybrid architecture CNN-BLSTM for automatic speech recognition," *AIoT Conference*, 2024.
[10] Jiabin Xue et al., "Exploring attention mechanisms based on summary information for end-to-end ASR," *Neurocomputing*, 2021.
[11] D. Toledano et al., "Multi-resolution speech analysis for ASR using deep neural networks," *PLOS ONE*, 2018.
[12] J. Chorowski et al., "Attention-based models for speech recognition," *arXiv*, 2015.
[13] J. Xue et al., "Structured sparse attention for end-to-end ASR," *IEEE ICASSP*, 2020.
[14] A. Graves, "Sequence transduction with recurrent neural networks," *ICML*, 2012.
[15] J. S. Garofolo et al., "TIMIT Acoustic-Phonetic Continuous Speech Corpus," *Linguistic Data Consortium*, 1993.
[16] L. Dong, J. Bai, J. Chen, and others, "Seed-ASR: Understanding Diverse Speech and Contexts with LLM-based Speech Recognition," *arXiv preprint arXiv:2407.04675*, 2024.
[17] W. Yu, C. Tang, G. Sun, and others, "Connecting Speech Encoder and Large Language Model for ASR," *ICASSP 2024–2024 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE, 2024.
[18] Li, Qijia, Zhang, Chao, and Woodland, Philip C., "Combining Hybrid DNN-HMM ASR Systems with Attention-Based Models Using Lattice Rescoring," *Speech Communication*, vol. 151, pp. 58–67, 2022. DOI: 10.1016/j.specom.2022.12.002.
[19] Gong, Xun, Wang, Wei, Shao, Hang, Chen, Xie, and Qian, Yanmin, "Factorized AED: Factorized Attention-Based Encoder-Decoder for Text-Only Domain Adaptive ASR," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4351–4355, 2023. DOI: 10.1109/ICASSP49357.2023.10095937.
[20] Zhang, Yu, Han, Wei, Qin, James, Wang, Yongqiang, and Bapna, Ankur, "Google USM: Scaling Automatic Speech Recognition Beyond 100 Languages," *arXiv preprint arXiv:2303.01037*, 2023.
[21] Yao, Zengwei, Guo, Liyong, Yang, Xiaoyu, Kang, Wei, Kuang, Fangjun, Yang, Yifan, Jin, Zengrui, Lin, Long, and Povey, Daniel, "Zipformer: A faster and better encoder for automatic speech recognition," *ICLR 2024 Conference Proceedings*, 2024. URL: https://github.com/k2-fsa/icefall.