Speech Recognition Using Convolutional and Recurrent Neural Networks

Muthanna Hasan
Department of Computer Science
Brock University
St. Catharines, Ontario, Canada
Student ID: 7518459
Email: ma21yb@brocku.ca

Alaa Masaoud
Department of Computer Science
Brock University
St. Catharines, Ontario, Canada
Student ID: 7774565
Email: rh22cg@brocku.ca

Ansh Malaviya
Department of Computer Science
Brock University
St. Catharines, Ontario, Canada
Student ID: 7729346
Email: db22rh@brocku.ca

***Abstract*—Automatic Speech Recognition (ASR) is a challenging learning problem due to the high variability present in speech signals, including differences in accents, speaking styles, and more. In this project our aim is to design and evaluate an end-to-end deep learning model for speech recognition, which combines a convolutional neural network (CNN), a recurrent neural network (RNN), and a connectionist temporal classification (CTC). The CNN component extracts local time-frequency features, while a bidirectional RNN captures long-range temporal dependencies across a sentence or a sequence of speech. A CTC-based classifier helps us use alignment-free training because we can map the variable length to the audio inputs directly to the character level.**

The model is implemented in PyTorch and trained on a large-scale dataset Librispeech. Performance is evaluated using training loss, validation loss, and word error rate (WER). Experimental results show that the model's architecture is capable of learning meaningful representations of words and is able to produce an increasingly accurate transcription of the words over time. This work highlights the effectiveness of end-to-end automatic speech recognition pipelines and provides insight into the trade-offs between model complexity, training stability, and recognition accuracy.

I. Introduction

Speech recognition is a common problem in artificial intelligence, it is very common, as a lot of the AI we see when we pick up the phone, etc., are trained to recognize your voice. For example, if you were to call RBC, they would have a robot that you speak to, and it would transcribe what you say over the phone to help you get the help you need. More applications of these machines that transcribe language into text are voice assistants such as Siri, text-to-speech agents, and more. Despite decades of research ASR still remains very difficult due to the high variability in speech, from everyone having different voices to accents, pronunciations, etc.

Traditional ASR systems used to rely on modular pipelines involving hand-crafted acoustic features, pronunciation lexicons, and probabilistic functions, such as the hidden markov model and the gaussian mixture model. These approaches work, but they require a lot of manual engineering. Recently, deep learning has been able to speed up and improve the development of the end-to-end ASR as they learn feature representation and sequence mapping directly from the data, allowing us to skip the manual engineering.

In this project we explore an end-to-end speech recognition model that integrates a convolutional neural network, a recurrent neural network, and a connectionist temporal classification. The main goal is to design, implement, and test a neural architecture that can convert speech signals into text without losing anything or requiring manual engineering. By using CNNs, RNNs, and CTCs, the system can reflect the core idea behind these modern ASR systems.

The model is implemented using the PyTorch learning framework and is trained on a large dataset, specifically LibriSpeech. The model performance is evaluated using training loss, validation loss, and word error rate (WER), providing insight into both recognition accuracy and computational efficiency.

II. Background Information

A. Speech Recognition as a Learning Problem

Speech recognition can be designed as a sequence-to-sequence learning problem, where an input sequence of observations is mapped to an output sequence of symbols. A key problem lies in the fact that the input and output sequences are different lengths, and the correspondence between them is unknown. Unlike other applications such as image classification, speech recognition requires models that can handle variable-length sequences and long-range dependencies.

Signals are typically transformed into time-frequency representations. These representations preserve both temporal and spectral information, making them good for neural network processing. However, learning directly from these representations to text requires models that can capture both local structure and broader temporal context.

B. End to End ASR

End-to-end ASR systems address this challenge by jointly learning feature extraction, temporal modelling, and decoding within a single neural network. Architectures combining CNNs, RNNs, and CTC have been proven to work, as they have a strong baseline due to their effectiveness and stability.

C. Neural Network Components

Convolutional neural networks are an important part of modern speech recognition systems by serving as feature extractors. In ASR applications, CNNs operate on two-dimensional representations, treating time and frequency as spatial dimensions. Convolutional filters learn to detect local spectral patterns.

By stacking multiple convolutional layers, the network learns hierarchical feature representations, where low-level features are gradually transformed into higher-level patterns that are useful for phoneme and character recognition. CNNs also offer stability to small variations in time and frequency, which commonly occur in speech signals.

GELU (Gaussian Error Linear Unit) activations are used throughout the network in place of traditional ReLU functions. Unlike ReLU, which applies a hard threshold, GELU weights inputs by their probability under a standard normal distribution, resulting in a smoother, non-linear activation. This smoothness improves gradient flow and has been shown to stabilize training in deep neural networks. GELU is commonly used in modern deep learning architectures and was selected to improve training stability and representational capacity in both convolutional and recurrent layers.

Speech is inherently sequential, with strong temporal dependencies spanning multiple time steps. Recurrent neural networks are designed to process sequential data by maintaining a hidden state that evolves over time. This allows information from earlier frames to influence predictions at later frames. Helping the network interpret the correct sequence.

Variants such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) mitigate the vanishing gradient problem, enabling the model to capture long-range dependencies. Bidirectional recurrent networks further enhance performance by processing the sequence in both forward and backward directions, allowing the model to leverage both past and future context.

In this project, the recurrent component receives the sequence of features produced by the CNN and models their temporal relationships. The output consists of contextualized hidden states at each time step, which encode both types of information necessary for accurate transcription.

D. Training, Data and Evaluation Considerations

One of the biggest challenges in ASR is the lack of explicit alignment between audio frames and output symbols. Connectionist Temporal Classification addresses this issue by providing a loss function that enables sequence-to-sequence training without frame-level supervision.

CTC introduces a special blank symbol and defines a many-to-one mapping from frame-level predictions to output sequences. During training, the loss function considers all valid alignments between the input and output and computes the total probability of producing the correct transcription. This approach eliminates the need for forced alignment or segmented training data.

In the implemented system, a CTC classifier maps recurrent hidden states to log probabilities over the target alphabet. The classifier consists of a feedforward network followed by a log-softmax operation, producing the log probabilities required by the CTC loss function.

The experiments conducted in this project utilize the LibriSpeech dataset. LibriSpeech consists of read English speech derived from audiobooks, with recordings. The dataset contains multiple subsets that vary in size and difficulty.

LibriSpeech is commonly used to evaluate ASR systems due to its standardized structure and public availability. The dataset is divided into training and validation splits, enabling consistent evaluation of generalized performance. In this project, the dataset is used to train the end-to-end speech recognition model that maps input sequences directly to character-level transcriptions.

The implemented system follows end-to-end architecture in which feature extraction, temporal modelling, and sequence prediction are learned jointly. Acoustic features are first extracted from raw audio signals and passed through the CNN to produce a sequence of learned representations. These representations are then processed by the recurrent network to understand and save long-range temporal dependencies.

The recurrent outputs are projected into a character-level output space using a CTC classifier, which produces log-probabilities over the target alphabet at each time step. During training, the CTC loss function computes the likelihood of the correct transcription by summing over all valid alignments. This design enables the model to handle inputs and outputs of different lengths without explicit alignment.

In addition to loss-based metrics, word error rate is computed on the validation set to evaluate transcription quality. Epoch-level training time is also recorded, providing insight into computational efficiency and scalability.

Word error rate is used as the primary metric for evaluating recognition performance. WER measures the number of changes required to transform a predicted transcription into the reference transcription, normalized by the number of words in the reference. This metric closely aligns with human perception of transcription quality.

Tracking both loss and WER provides a comprehensive view of model performance, as improvements in optimization do not always translate directly into improved recognition accuracy.

Prior to being processed by the neural network, raw audio waveforms are transformed into time-frequency representations that are more suitable for learning. Speech signals are non-stationary, meaning their frequency content changes over time.

These representations preserve both temporal resolution and frequency information, enabling convolutional filters to learn meaningful local patterns. By operating on structured acoustic features rather than raw waveforms, the model benefits from a representation that emphasizes phonetic content while reducing sensitivity to minor signal variations, helping stability.

The extracted features are normalized to improve numerical stability during training. Feature normalization helps ensure that input values remain within a consistent range, facilitating faster convergence and reducing sensitivity to initialization and learning rate selection.

The speech recognition system in this project performs prediction at the character level rather than the word level. Character-level modelling offers several advantages in end-to-end ASR systems. Unlike word-level modelling, it does not require a predefined vocabulary or pronunciation, allowing the model to generalize to previously unseen words.

Character-level outputs also integrate naturally with the CTC framework, which is well suited for predicting sequences of discrete symbols without explicit alignment. By predicting characters directly, the model learns implicit representations of phonetic and linguistic structure while maintaining flexibility in decoding.

During inference, the model produces a sequence of probability distributions over the character alphabet at each time step. These frame-level predictions must be converted into a final textual transcription.

The resulting sequence is post-processed by collapsing repeated characters and removing blank symbols, yielding the predicted transcription. While greedy decoding is computationally efficient and simple to implement, it does not account for alternative alignments or linguistic constraints.

More advanced decoding strategies such as beam search and language model integration can further improve recognition accuracy. However, greedy decoding provides a reasonable baseline and allows the performance of the acoustic model itself to be evaluated without additional external components.

Deep neural networks for speech recognition are prone to overfitting due to the large number of parameters and the complexity of speech data.

Validation metrics are monitored throughout training to assess generalized performance. By evaluating the model on held-out data, the training process can be adjusted to prevent excessive overfitting and ensure that learned representations generalize beyond the training set.

Training end-to-end speech recognition models is computationally intensive due to long input sequences and large model sizes. The use of convolutional layers helps reduce sequence length and computational cost by downsampling the input representations.

Epoch-level training time is recorded as part of the evaluation process to quantify computational efficiency. This metric provides insight into the scalability of the model and highlights trade-offs between recognition accuracy and training cost.

Efficient implementation using optimized deep learning libraries is essential for managing the computational demands of ASR systems. PyTorch provides automatic differentiation and GPU acceleration.

While the CNN-RNN-CTC architecture is effective, it also presents several limitations. The reliance on CTC assumes conditional independence between output predictions, which may limit the model's ability to capture strong linguistic dependencies. Additionally, character-level modelling may require longer sequences to represent words, increasing decoding complexity.

Although we have these limitations, the architecture provides a strong baseline for end-to-end speech recognition and serves as a foundation for future improvements.

III. Experimental Setup

A. Dataset

All experiments were conducted using the *LibriSpeech* dataset, which is a standard benchmark for automatic speech recognition tasks. LibriSpeech consists of read English speech derived from audiobooks and provides high-quality audio-text alignments.

For this project, two official dataset splits were used:

- **train-clean-100**, which was used for training the model
- **test-clean**, which was used exclusively for evaluation

These splits were selected as they contain clean speech recordings with minimal background noise. This allows the model's learning behaviour and convergence to be analyzed without additional noise-related factors influencing performance.

The task is formulated as a *character-level* speech recognition problem, meaning that the model predicts individual characters rather than full words or subword units. Each audio input is mapped to a sequence of characters, and complete words are formed implicitly through the decoding process.

The output vocabulary consists of 28 symbols, including a blank token required for CTC training, the space character, and the 26 uppercase English letters (A-Z). All transcripts were converted to uppercase, and any characters outside this vocabulary were removed to ensure consistency during training and decoding.

Due to the size of the dataset (approximately 7 GB for the selected splits), the raw audio files were not included in the project repository. Instead, reproducibility is ensured through the use of the official LibriSpeech dataset and a provided preprocessing pipeline that converts the raw audio into the exact processed format used in the experiments. This approach keeps the repository manageable while still allowing the full experimental setup to be recreated.

B. Data Preprocessing

All audio data was preprocessed prior to training in order to standardize the input format and reduce variability in the raw recordings. The preprocessing pipeline was designed to be fully reproducible and to produce variable-length sequences padded at batch time suitable for CTC-based sequence learning.

Each audio file was first resampled to a sampling rate of 16 kHz. This ensured consistent temporal resolution across all recordings, as the original LibriSpeech files may vary in sample rate. The resampled waveforms were then converted into log-Mel spectrogram representations, which provide a compact time–frequency description of the speech signal. The following parameters were used for feature extraction: 128

Mel filter banks, a Fast Fourier Transform size of 1024, and a hop length of 160 samples. The resulting spectrograms were converted to the decibel scale and normalized using per-sample mean and variance normalization.

Corresponding transcript files were processed in parallel with the audio. All text was converted to uppercase, and any characters not present in the predefined vocabulary were removed. The cleaned transcripts were then encoded as sequences of integer indices, where each index corresponds to a character in the vocabulary.

To improve data loading efficiency during training, each processed sample was saved individually as a serialized `.pt` file. Each file contains the log-Mel spectrogram features, the encoded transcript, and the corresponding input and target sequence lengths. To reduce disk usage and I/O overhead, spectrogram features were stored in half precision (`float16`) on disk and converted back to single precision (`float32`) when loaded for training.

During batching, a custom collation function was used to handle variable-length inputs. Spectrograms were padded along the time dimension to match the longest sequence in each batch, while target sequences were concatenated without padding. The original input and target lengths were preserved and passed directly to the CTC loss function to ensure correct alignment during training.

C. Model Architecture

The proposed system follows an end-to-end architecture for automatic speech recognition. The model processes log-Mel spectrogram inputs and produces character-level transcriptions using a combination of convolutional feature extraction, recurrent sequence modelling, and Connectionist Temporal Classification (CTC). The overall pipeline consists of four main components: a convolutional neural network (CNN), a fully connected projection layer, a bidirectional recurrent neural network (GRU), and a CTC-based classifier.

Figure 1 shows an overview of the end-to-end speech recognition architecture used in this work.

a) Convolutional Feature Extractor: The CNN operates directly on the log-Mel spectrogram inputs, which are treated as two-dimensional feature maps. The network consists of three convolutional layers, each using a kernel size of 3×3 and a base channel size of 32. To reduce the temporal and spectral resolution early in the pipeline, a stride of $(2, 2)$ is applied in the first convolutional layer, while all subsequent layers use a stride of $(1, 1)$. Each convolutional block includes batch normalization, GELU activation, and dropout for regularization.

Residual connections are applied when the input and output tensor dimensions of a convolutional block match. These connections allow information to pass directly through the network and help maintain stable gradient flow during training. The output of the CNN is a higher-level representation of the input spectrogram with reduced temporal resolution.

b) Feature Projection: The output of the CNN has the shape $[B, C, F, T]$, where B denotes the batch size, C the number of channels, F the frequency dimension, and T the time dimension. This tensor is permuted to $[B, T, C, F]$ and flattened across the channel and frequency dimensions to produce a feature vector for each time step. A fully connected layer then projects this flattened representation to a 512-dimensional feature space, which serves as the input to the recurrent network.

To ensure flexibility when modifying the CNN architecture or preprocessing parameters, the dimensionality of the CNN output is inferred dynamically by passing a dummy input through the convolutional layers during model initialization.

c) Recurrent Network: Temporal modelling is performed using a three-layer bidirectional Gated Recurrent Unit (GRU) network. Each GRU layer has a hidden size of 512 units per direction, resulting in a 1024-dimensional output at each time step. Layer normalization and GELU activation are applied after each recurrent layer, followed by dropout for regularization. Residual connections are used when the input and output dimensions of a layer are compatible.

The bidirectional structure allows the model to incorporate both past and future context when generating predictions, which is particularly important for speech recognition tasks where phonetic and linguistic cues depend on surrounding audio.

d) CTC Classifier: The output of the recurrent network is passed to a CTC-based classifier. This classifier consists of a small feedforward network composed of a linear layer, GELU activation, dropout, and a final linear projection to the output vocabulary size. The vocabulary contains 28 symbols, including a blank token, the space character, and the 26 uppercase English letters. Log-softmax is applied along the class dimension to produce log-probabilities required by the CTC loss function.

e) Sequence Length Handling: Since the CNN reduces the temporal resolution of the input through striding, the effective output sequence length differs from the original input length. The output lengths are computed by dividing the original input lengths by the CNN stride and clamping the result to the actual size of the model output. These lengths are passed explicitly to the CTC loss to ensure correct alignment between the predicted sequences and target transcripts during training.

D. Training Procedure

Training was performed in an end-to-end manner using Connectionist Temporal Classification (CTC) loss. CTC enables sequence-to-sequence learning without requiring explicit frame level alignments between the audio input and the target transcription, making it well suited for speech recognition tasks with variable-length inputs.

The model was optimized using the Adam optimizer with an initial learning rate of 1×10^{-3}. To maintain training stability, gradient norms were clipped to a maximum value of 5, which helps prevent exploding gradients that can arise when training deep recurrent architectures on long speech sequences.

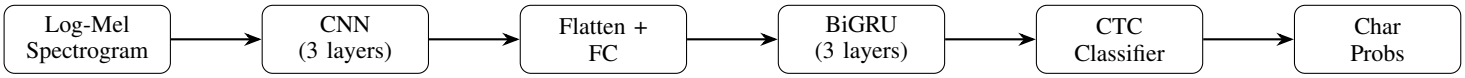

Fig. 1. Overview of the end-to-end speech recognition pipeline. Log-Mel spectrograms are processed by a convolutional feature extractor, projected to a fixed-dimensional representation, modeled temporally using bidirectional recurrent layers, and decoded using a CTC-based classifier.

```
optimizer = Adam(model.parameters(),
lr = LEARNING_RATE)
```

Due to memory constraints associated with processing long audio samples, gradient accumulation was employed. A batch size of 32 samples was used, and gradients were accumulated over four consecutive batches before performing a parameter update. This effectively simulates a larger batch size of 128 while remaining within GPU memory limits.

Mixed-precision training was enabled using automatic casting and gradient scaling. This allows selected operations to be executed in half precision, reducing memory usage and improving computational efficiency, while gradient scaling prevents numerical instability during backpropagation.

Model checkpoints were saved after each epoch to allow training to be resumed and to enable analysis of intermediate model states. A learning rate scheduler based on validation loss was used to reduce the learning rate when performance plateaued. Early stopping was applied to terminate training when the validation loss failed to improve for a fixed number of consecutive epochs.

Training was conducted on GPU hardware with support for CUDA or ROCm acceleration. Additional memory management steps, such as explicit cache clearing, were used to ensure stable training during long experimental runs.

E. Evaluation Method

Evaluation was performed on the *test-clean* split of the LibriSpeech dataset, which was not used during training. This split contains clean speech recordings and is commonly used for assessing baseline speech recognition models.

During evaluation, the model was run in inference mode with gradient computation disabled. For each input utterance, the model produced a sequence of character probabilities over time. These probabilities were decoded using greedy CTC decoding. At each time step, the most likely character was selected, repeated characters were collapsed, and blank symbols were removed to form the final predicted transcript.

Model performance was measured using Word Error Rate (WER). WER compares the predicted transcript with the ground-truth text and measures the number of word-level insertions, deletions, and substitutions required to convert one into the other, normalized by the total number of words in the reference transcript. This metric is widely used in speech recognition and provides an intuitive measure of transcription accuracy.

In addition to WER, the CTC loss on the test set was recorded after each training epoch. Tracking both loss and WER helps show how training progress relates to actual transcription quality.

To better understand model behaviour, a small number of test samples were also inspected at selected training checkpoints. For these samples, the predicted transcripts were compared directly to the ground-truth text to observe how output quality changed over time.

IV. Results

This section presents the quantitative and qualitative results obtained from training and evaluating the proposed speech recognition model on the LibriSpeech dataset. Model performance is analyzed in terms of training behaviour, recognition accuracy, and example transcriptions.

A. Training Behavior

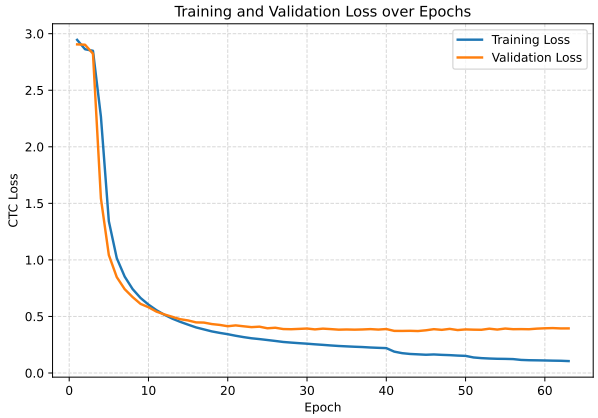

Fig. 2. Training and validation CTC loss over epochs. Both losses decrease rapidly during early training and stabilize as the model approaches convergence.

Figure 2 shows the training and validation CTC loss over the course of training. The training loss decreases steadily from approximately 2.95 in the first epoch to below 0.11 by epoch 63, indicating that the model successfully learns to align audio inputs with character sequences.

The validation loss follows a similar downward trend, decreasing rapidly during the early epochs and then stabilizing as training progresses. After approximately 40 epochs, improvements in validation loss become more gradual, suggesting that the model approaches convergence. Minor fluctuations in validation loss at later epochs are expected due to the complexity of long-form speech sequences and do not indicate training instability.

Each training epoch required approximately 17 to 18 minutes to complete, reflecting the computational cost of training a deep end-to-end speech recognition model on long audio sequences. Due to this cost, training was stopped once convergence behaviour was observed.

B. Recognition Performance

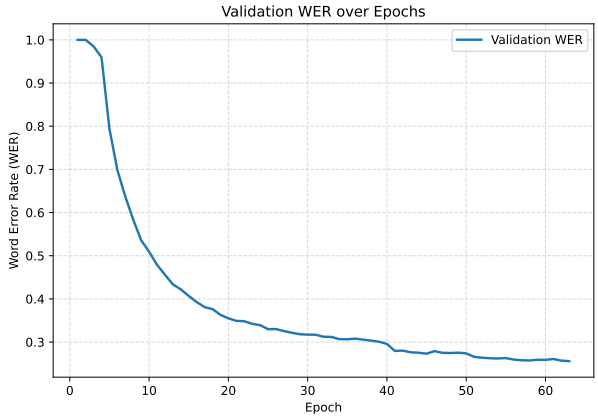

Fig. 3. Validation Word Error Rate (WER) across training epochs. WER decreases steadily as training progresses and stabilizes in later epochs.

Model performance was evaluated using Word Error Rate (WER) on the *test-clean* split. Figure 3 illustrates the validation WER across training epochs. Early in training, the WER remains close to 1.0, indicating that the model initially fails to produce meaningful transcriptions.

As training progresses, WER decreases steadily, dropping below 0.5 by epoch 10 and continuing to improve until stabilizing near 0.26–0.28 in later epochs. The best observed validation WER is approximately 0.26, achieved after extensive training. This trend demonstrates that reductions in CTC loss correspond directly to improvements in transcription accuracy.

While the final WER remains higher than that of large-scale production systems, the result is reasonable given the limited dataset size, character-level modelling approach, and the absence of an external language model during decoding.

C. Qualitative Transcription Analysis

TABLE I
EXAMPLE TRANSCRIPTIONS AT DIFFERENT TRAINING STAGES

Checkpoint	Target Transcript	Predicted Transcript	WER
Epoch 2	WELL AS I SAY ITS AN AWFUL QUEER WORLD …	(empty output)	1.00
Epoch 11	WELL AS I SAY ITS AN AWFUL QUEER WORLD …	WELL AS I SAY IS AT AFL QUER WORLD …	0.61
Final	WELL AS I SAY ITS AN AWFUL QUEER WORLD …	WELL AS I SAY ITS AN AWFUL QUEER WORLD …	0.40

To complement the quantitative metrics, several test samples were examined at different stages of training. Table I shows example transcriptions that were analyzed from an early checkpoint (epoch 2), a mid-training checkpoint (epoch 11), and the final trained model.

At epoch 2, the model fails to generate meaningful output, producing empty or near-empty predictions with a WER of 1.0. By epoch 11, the model begins to capture phonetic structure, producing outputs that resemble the target text but contain frequent spelling errors, substitutions, and omissions.

In the final model, predictions closely resemble the ground-truth transcripts, with most words correctly identified. Remaining errors primarily involve minor spelling mistakes, homophone substitutions, or inaccuracies in rare words. Despite these errors, the overall sentence structure and semantic content are preserved, demonstrating that the model has learned effective acoustic-to-text mappings.

These qualitative observations align with the steady improvement in WER and confirm that transcription quality continues to improve throughout training.

V. CONCLUSION

This project presented the design, training, and evaluation of an end-to-end speech recognition system trained on the LibriSpeech dataset. The model combines convolutional layers for acoustic feature extraction, bidirectional recurrent layers for temporal modelling, and a CTC-based objective to enable alignment-free transcription at the character level.

Experimental results show that the model learns effectively from raw log-Mel spectrogram inputs. Training and validation loss decrease steadily, and validation Word Error Rate improves consistently across epochs, reaching a final WER of approximately 0.26 on the *test-clean* split. Qualitative analysis of predicted transcriptions further confirms that the model progressively captures phonetic structure and produces increasingly accurate and readable outputs as training progresses.

The model demonstrates stable convergence and meaningful transcription capability using a relatively compact and interpretable architecture. Overall, this paper demonstrates the feasibility and effectiveness of end-to-end neural approaches for automatic speech recognition.

REFERENCES

[1] Prof. D. Bockus, *COSC 4P80: Introduction to Artificial Neural Networks – Lecture Slides*, Department of Computer Science, Brock University, 2025. Available on Brightspace.

[2] PyTorch, *Audio Transforms Documentation*, pytorch.org. Available: https://docs.pytorch.org/audio/stable/transforms.html

[3] PyTorch, *Torchaudio Documentation*, pytorch.org. Available: https://pytorch.org/audio/stable/index.html

[4] Python Software Foundation, *Python 3 Documentation*, python.org. Available: https://docs.python.org/3/

[5] CodeGenes, *Working with LibriSpeech in PyTorch*, codegenes.net. Available: https://www.codegenes.net/blog/librispeech-pytorch/

[6] Real Python, *Python Torchaudio: A Practical Guide*, realpython.com. Available: https://realpython.com/python-torchaudio/#sampling

[7] PyTorch, *Audio Feature Extractions Tutorial*, GitHub Repository. Available: https://github.com/pytorch/audio/blob/main/examples/tutorials/audio_feature_extractions_tutorial.py

[8] AssemblyAI, *End-to-End Speech Recognition in PyTorch*, assemblyai.com. Available: https://www.assemblyai.com/blog/end-to-end-speech-recognition-pytorch

[9] Keras Documentation, *Automatic Speech Recognition using CTC*, keras.io. Available: https://keras.io/examples/audio/ctc_asr/

APPENDIX

This appendix presents full-sized versions of both graphs (losses and WER). These figures are included for improved readability. Each supplementary figure corresponds directly to the figure referenced in the Results section.

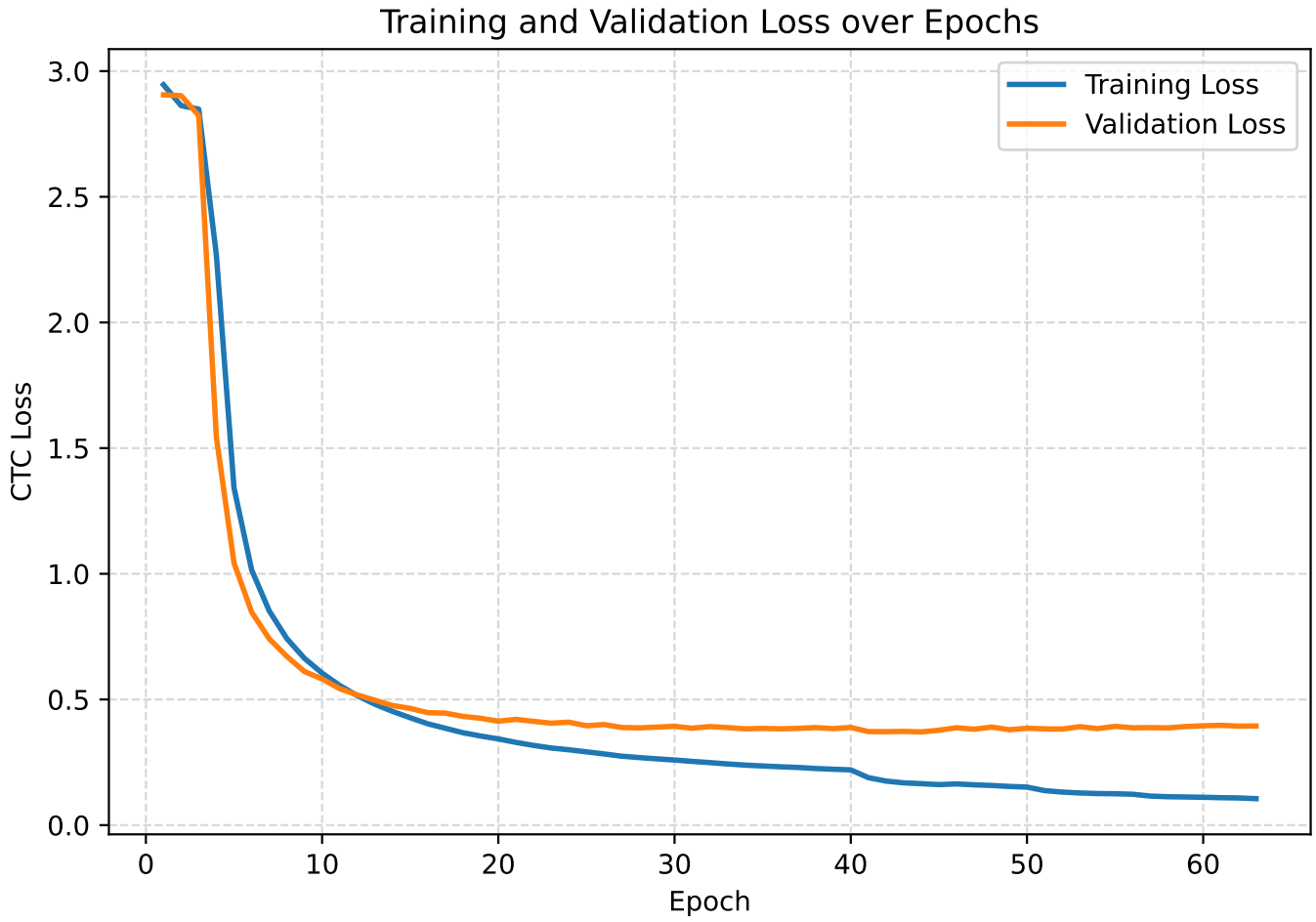

Fig. 4. Supplementary version of Figure 2: Training and validation CTC loss over epochs.

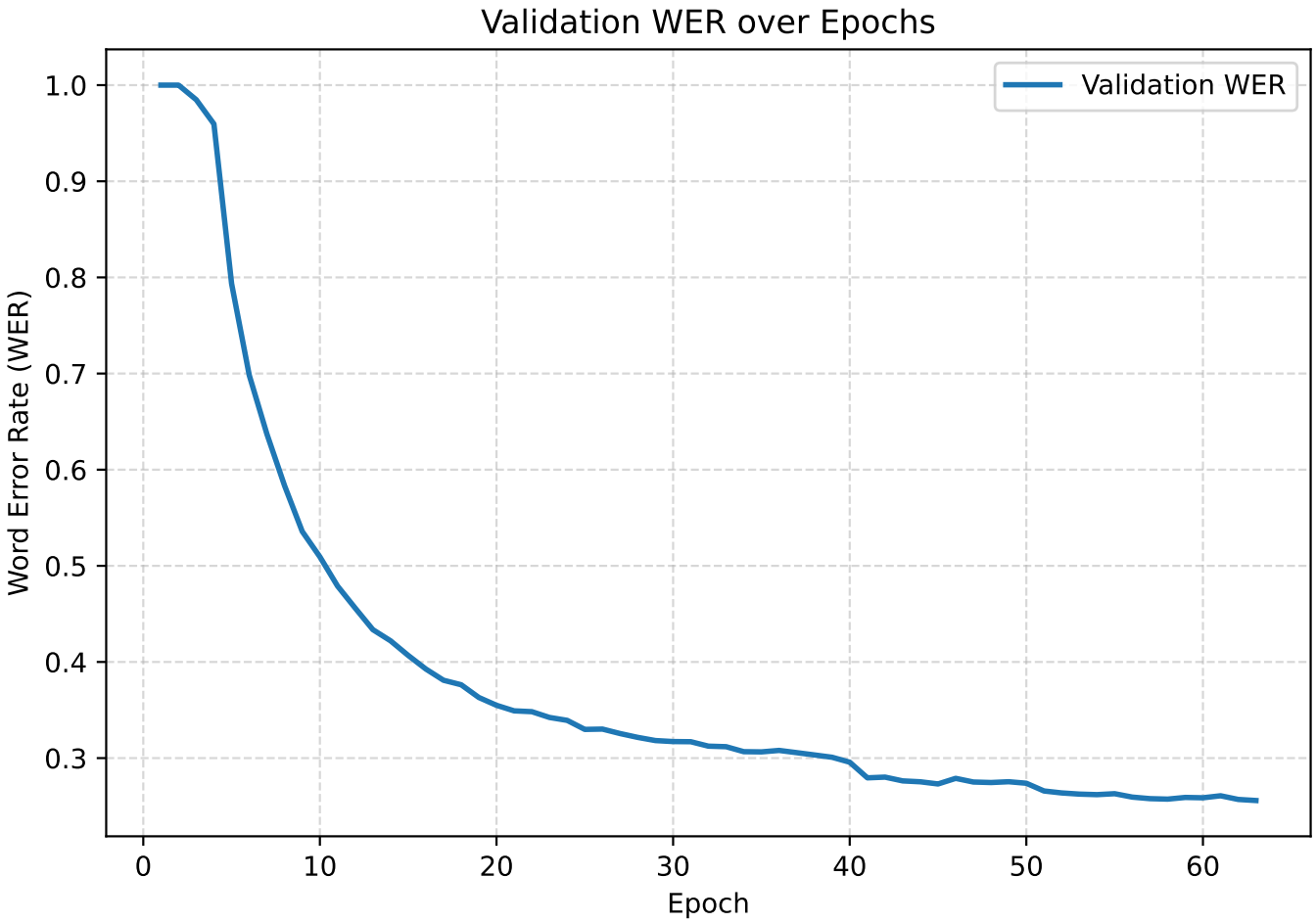

Fig. 5. Supplementary version of Figure 3: Validation Word Error Rate (WER) across training epochs.