End-to-End Bengali Speech Transcription

CSE 465.4, Group 4

Shakhawat Hossain
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
Email: shakhawat.hossain1@northsouth.edu

Ashfaqur Rahman Chowdhury
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
Email: ashfaqur.chowdhury@northsouth.edu

***Abstract*—This paper focuses on developing an efficient and accurate automatic speech recognition (ASR) system for the Bangla language. Despite advancements in multilingual ASR models, transcribing Bangla speech accurately remains a challenge due to limited resources and higher Word Error Rates (WER). In this project, we implement a Connectionist Temporal Classification (CTC)-based model designed and trained from scratch, optimized specifically for Bangla audio transcription. Additionally, we fine-tune some transformer-based ASR models, such as Whisper and FastConformer, on Bangla datasets. Finally, we conduct a comparative analysis of the performance of these models and analyze their relative WER to contribute to the improvement of open-source ASR tools for Bangla. This paper discusses the evolution of ASR systems, our custom system design, the experimental results, and the implications of our findings.**

***Index Terms*—Automatic Speech Recognition, Bangla Speech-to-Text, Connectionist Temporal Classification, Transformer ASR, Word Error Rate**

I. Introduction

Automatic Speech Recognition (ASR) systems have seen significant advancements in recent years, driven by the development of end-to-end deep learning approaches. While models like Whisper have shown great success in transcribing multilingual audio, Bangla ASR remains a relatively underexplored area, with existing systems reporting higher Word Error Rates (WER) compared to other widely spoken languages.

The core contribution of this paper involves:

- Providing a comprehensive overview of the evolution of ASR architectures, including CTC-based, self-supervised, and hybrid models.
- Developing a custom CTC-based ASR model designed and trained from scratch for Bangla audio.
- Fine-tuning a state-of-the-art transformer-based ASR model, such as Whisper, on Bangla datasets to optimize their performance for Bangla transcription.
- Conducting a comparative analysis of the WER across different architectures, providing insights into the suitability of each approach for Bangla ASR tasks.

By the end of this project, we aim to present two to three models: a CTC-based model trained from scratch and one or two fine-tuned transformer-based models, along with a comprehensive performance comparison.

II. Literature Review

A. CTC-Based Models

Connectionist Temporal Classification (CTC) [1] is a loss function designed for sequence-to-sequence tasks like speech recognition, where the alignment between input audio and output text is unknown. CTC-based models are particularly effective for ASR because they eliminate the need for frame-level alignment, enabling end-to-end learning.

DeepSpeech2 [2], developed by Baidu in 2015, exemplifies the use of CTC in ASR. Its architecture consists of recurrent neural networks (RNNs) for capturing the temporal dependencies in audio sequences, followed by a softmax layer that predicts character probabilities at each time step. The model incorporates batch normalization to accelerate convergence and improve generalization, especially in noisy and diverse environments. A key feature of DeepSpeech2 is its ability to transcribe both English and Mandarin using the same framework, showcasing its robustness across languages. However, its reliance on external language models to handle large vocabularies and improve fluency remains a limitation. Without such language models, the generated text may lack coherence, particularly in large vocabulary applications.

Despite their strengths, CTC-based models have limitations in capturing long-range dependencies and tend to struggle with out-of-vocabulary words. These issues have driven the development of more advanced architectures, such as transformer-based models.

B. Self-Supervised Transformer Models

Self-supervised learning has revolutionized ASR by leveraging unlabeled audio data for pre-training, significantly reducing the dependency on large labeled datasets. Models like Facebook's wav2vec 2.0 [3] and HuBERT [4] represent milestones in this domain.

Wav2vec 2.0 employs a convolutional feature encoder to extract latent audio representations, which are then processed by a transformer encoder. During pre-training, parts of the input audio are masked, and the model is trained to predict the masked sections using contrastive loss. This masking strategy is inspired by the masked language modeling techniques used in natural language processing, allowing the model to learn

rich, contextual representations of speech. Fine-tuning the model on labeled data enables it to excel in low-resource scenarios, achieving state-of-the-art results on several benchmarks.

HuBERT builds upon wav2vec 2.0 by introducing a clustering-based approach for generating targets during pre-training. Instead of relying on the convolutional encoder's output, HuBERT uses embeddings from intermediate layers of a transformer encoder, which are refined over multiple iterations. This iterative process improves the quality of the learned representations, resulting in better performance on downstream tasks. HuBERT also simplifies the training process by using cross-entropy loss, as opposed to wav2vec 2.0's more complex contrastive and diversity losses.

Both models have significantly improved ASR performance, particularly in low-resource settings. However, their reliance on large-scale computing resources for pre-training and fine-tuning poses challenges for smaller-scale deployments.

C. Weakly-Supervised Transformer Models

Weakly-supervised models aim to generalize ASR systems across languages and domains by leveraging large-scale datasets with varying levels of transcription quality. These models often adopt an encoder-decoder transformer architecture, which is well-suited for handling sequence-to-sequence tasks.

Whisper [5], developed by OpenAI in 2022, exemplifies this approach. Trained on 680k hours of labeled audio spanning 96 languages, Whisper uses weak supervision to achieve robust transcription capabilities across diverse accents, noise conditions, and languages. Its architecture includes an encoder that processes the input audio into a series of latent representations, followed by a decoder that generates the transcription. Whisper's multitask training setup allows it to perform tasks like speech-to-text translation and language detection in addition to transcription.

One of Whisper's notable strengths is its zero-shot capability, enabling it to transcribe languages and accents it has not been explicitly trained on. This makes it particularly valuable for underrepresented languages like Bangla. However, its reliance on massive datasets and significant computational resources for training limits its accessibility for smaller organizations or individual researchers.

D. Hybrid Architectures (Convolution + Self-Attention)

Hybrid architectures combine the strengths of convolution and self-attention mechanisms to capture both local and global dependencies in audio sequences. This blend allows for efficient modeling of short-term acoustic patterns while maintaining the capacity to understand long-range contextual relationships.

Conformer [6], developed by Google in 2020, integrates convolutional layers for extracting fine-grained acoustic features and self-attention layers for capturing global context. Conformer achieves state-of-the-art results on benchmarks like LibriSpeech [7], demonstrating its ability to handle diverse speech data effectively. However, the model's quadratic memory and computational complexity make it resource-intensive, limiting its scalability for very long audio sequences.

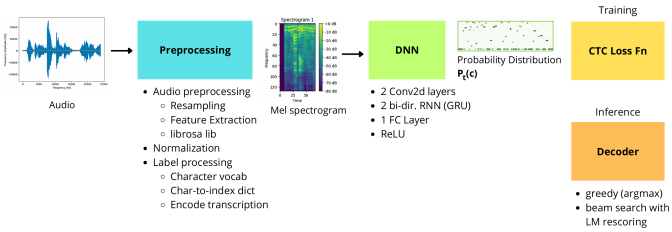

Fig. 1. Architecture of the custom CTC-based model

FastConformer [8], proposed by NVIDIA in 2023, addresses these limitations by introducing a linearly scalable attention mechanism and an optimized downsampling schema. By replacing global attention with limited-context attention and adding a global token for context aggregation, FastConformer significantly reduces memory usage and computational overhead. This makes it suitable for applications requiring real-time processing or handling longer audio segments. Despite its efficiency, FastConformer may compromise on fine-grained accuracy in scenarios requiring detailed transcription.

Hybrid architectures like Conformer and FastConformer represent the next step in ASR development, balancing accuracy and efficiency to meet diverse application needs.

III. Methodologies

A. System Design

Our project involves developing and comparing three Automatic Speech Recognition (ASR) models for Bangla speech transcription:

- Custom CTC-Based Model: Designed and trained from scratch using a Connectionist Temporal Classification (CTC) architecture optimized for Bangla.
- Fine-Tuned Whisper-Small Model [9]: OpenAI's transformer-based model, fine-tuned on the same Bangla dataset.

Both models share a unified preprocessing pipeline (from Figure 1) for consistency:

- Audio Conversion: All audio files are resampled to 16 kHz mono-channel.
- Feature Extraction: For Custom CTC Model: Mel-frequency cepstral coefficients (MFCCs) are used. For Transformer Models Log-Mel spectrograms are computed via model-specific feature extractors.
- Normalization: Features are normalized to zero mean and unit variance.
- Tokenization: In CTC, it is common to classify speech chunks into letters, so for the CTC-based models we have extracted all distinct letters from the datasets and built our vocabulary from this set of letters.
- Dataset Splitting: Dataset is split into training, validation, and test sets.

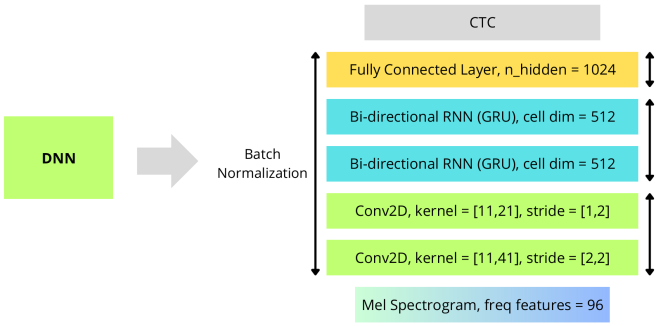

Fig. 2. Network design of the custom CTC-based model

As for the model architecture (shown in Figure 2), our custom CTC-based model is configured with the following components:

- Convolutional Layers: Two 2D convolution layers with kernel sizes $[11, 41]$ and $[11, 21]$, strides $[2, 2]$ and $[1, 2]$, and 32 channels each.
- Recurrent Layers: Two bidirectional GRU layers, each with 512 units per direction.
- Fully Connected Layer: A 1024-unit layer with ReLU activation.
- Output Layer: Softmax layer producing probabilities for each character in the vocabulary, including a blank token for CTC.
- Regularization: Dropout with keep probability of 0.5 and L2 regularization.
- Optimizer: Adam optimizer with exponential learning rate decay.

B. Required Software

The project is implemented using the following software tools:

- Google Colab pro+ for notebook and training hardware.
- Libraries and Frameworks:
  - PyTorch [10] for implementing the custom CTC model and fine-tuning.
  - Hugging Face Transformers library [11] for Whisper fine-tuning.
  - Librosa [12] and Torchaudio [13] for audio preprocessing.
  - JiWER for calculating Word Error Rate (WER). [14]
- Pretrained model:
  - Whisper-Small [9]

C. Dataset Description

We use the Bangla subset of the Mozilla Common Voice dataset (version 11) [15]

- Total Recorded Hours: 461 hours.
- Validated Hours: 64 hours.
- Number of Voices: 20,866.
- Features: path, audio, sentence (additional features are removed since we are considering just the transcribed text for training and fine-tuning).

IV. Results and Conclusion

A. Data Preprocessing Techniques

For both models, a unified preprocessing pipeline was used to ensure consistency in the data input:

- **Audio Conversion**: Audio files were resampled to a mono-channel 16 kHz sampling rate.
- **Feature Extraction**:
  - For the custom CTC-based model, Mel-frequency cepstral coefficients (MFCCs) were used.
  - For Whisper-Small, log-Mel spectrograms were computed using the Hugging Face feature extractor.
- **Normalization**: Features were normalized to have zero mean and unit variance.
- **Tokenization**: The CTC-based model used a vocabulary derived from unique characters in the dataset, while Whisper-Small used the pre-trained tokenizer.
- **Dataset Splitting**: The commonvoice dataset was already split into training, validation, and test sets.

B. Training and Testing

1) Custom CTC-Based Model: The DeepSpeech2-inspired model consisted of two 2D convolution layers, two bidirectional GRU layers, and a fully connected layer. Training was performed using a Connectionist Temporal Classification (CTC) loss function, optimized with the Adam optimizer. Dropout and L2 regularization were applied for generalization. Training converged across five epochs, achieving a Word Error Rate (WER) of 55.56% by the final epoch.

2) Whisper-Small Model: OpenAI's Whisper-Small was fine-tuned using a PEFT (Parameter-Efficient Fine-Tuning) [16] adapter named LoRA (Low-Rank Adaptation) [17], combined with 8-bit quantization [18]. Only 1.44% of the total parameters (3,538,944 trainable parameters) were updated during training. The fine-tuning process required approximately 2.5 hours over two epochs, achieving an evaluation WER of 39.96% and a normalized WER of 18.57%.

TABLE I
Whisper-Small LoRA Fine-Tuning Results

Epoch	Training Loss	Validation Loss	WER (%)
1	0.1789	0.1796	39.96
2	0.0910	0.1306	18.57

C. Results

- **Custom CTC-Based Model**: The WER improved steadily over epochs, with the model reaching WER of 55.56% after five epochs. However, the model required more training iterations and additional decoding methods, such as beam search with an external n-gram language model, for further improvements.
- **Whisper-Small Model**: Despite its limited trainable parameters, Whisper-Small demonstrated robust performance on the Bangla dataset after fine-tuning, showcasing strong generalization with significantly less training time.

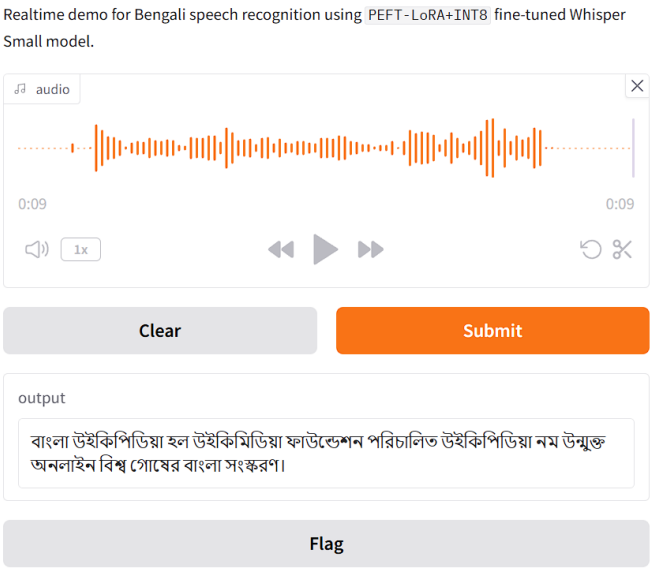

Fig. 3. Test Inference of the Finetuned Whisper-small using Gradio

D. Comparison

The table below compares the performance of the two models over training epochs:

TABLE II
COMPARISON OF WER (%) ACROSS MODELS

Epoch	Custom CTC Model	Whisper-Small
1	100.0	39.96
2	100.0	18.57
3	80.39	-
4	65.31	-
5	55.56	-

E. Conclusion

This study highlights the challenges and opportunities in developing ASR systems for the Bangla language. The Whisper-Small model demonstrated strong performance with limited fine-tuning, making it a robust choice for low-resource languages. In contrast, the custom CTC-based model, though capable of achieving promising results, required more extensive training and additional post-processing techniques for competitive performance.

Future research can explore advanced techniques such as quantization and further fine-tuning of transformer-based ASR models to enable efficient on-device offline usage. These approaches could significantly enhance the accessibility of ASR systems for Bangla and other low-resource languages.

REFERENCES

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

[2] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*, pp. 173–182, PMLR, 2016.

[3] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12449–12460, 2020.

[4] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 29, pp. 3451–3460, 2021.

[5] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International conference on machine learning*, pp. 28492–28518, PMLR, 2023.

[6] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[7] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210, IEEE, 2015.

[8] D. Rekesh, N. R. Koluguri, S. Kriman, S. Majumdar, V. Noroozi, H. Huang, O. Hrinchuk, K. Puvvada, A. Kumar, J. Balam, *et al.*, "Fast conformer with linearly scalable attention for efficient speech recognition," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8, IEEE, 2023.

[9] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022.

[10] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[11] T. Wolf, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[12] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python.," in *SciPy*, pp. 18–24, 2015.

[13] J. Hwang, M. Hira, C. Chen, X. Zhang, Z. Ni, G. Sun, P. Ma, R. Huang, V. Pratap, Y. Zhang, *et al.*, "Torchaudio 2.1: Advancing speech recognition, self-supervised learning, and audio processing components for pytorch," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–9, IEEE, 2023.

[14] A. C. Morris, V. Maier, and P. D. Green, "From wer and ril to mer and wil: improved evaluation measures for connected speech recognition.," in *Interspeech*, pp. 2765–2768, 2004.

[15] S. Alam, A. Sushmit, Z. Abdullah, S. Nakkhatra, M. Ansary, S. M. Hossen, S. M. Mehnaz, T. Reasat, and A. I. Humayun, "Bengali common voice speech dataset for automatic speech recognition," *arXiv preprint arXiv:2206.14053*, 2022.

[16] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, "Peft: State-of-the-art parameter-efficient fine-tuning methods." https://github.com/huggingface/peft, 2022.

[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[18] A. Bhandare, V. Sripathi, D. Karkada, V. Menon, S. Choi, K. Datta, and V. Saletore, "Efficient 8-bit quantization of transformer neural machine language translation model," *arXiv preprint arXiv:1906.00532*, 2019.