# A Bilingual Approach to Speech Recognition and Translation: Hindi to English Conversion

Debanjan Nanda* Debayan Datta† Ayan Maity‡
**VOXLING**

November 28, 2024

**Abstract**

The communication between native Hindi speakers and non-native Hindi speaking regions is a challenge. We develop a Hindi Speech to English Text Translation System that converts Hindi audio to Hindi text and then translates it into English. Our model is divided into two parts. The first part deals with Automatic Speech Recognition (ASR) and the second part works on Neural Machine Translation (NMT). This system aims to enhance accessibility for non-native Hindi speakers and facilitate communication for Hindi speakers in non-Hindi-speaking regions. By leveraging advanced speech recognition and machine translation technologies, we address real-time translation needs and advance research in low-resource languages.

## 1 Introduction

Language barriers often hinder effective communication and limit access to content, particularly for non-English speakers. To address this challenge, we designed a system that bridges the gap by converting Hindi audio into Hindi text and subsequently translating it into English. This approach not only facilitates smoother communication but also enables better access to Hindi content for English speakers and broader audiences.

Speech-to-text translation (ST) systems typically operate in two distinct steps: first, using automatic speech recognition (ASR) to transcribe the source language audio into text, and second, employing machine translation (MT) to translate the text from the source language into the target language. NMT aims to translate an input sentence from the source language to the target one. An NMT model usually consists of an encoder, a decoder and an attention module. The encoder maps the input sequence to hidden representations and the decoder maps the hidden representations to the target sequence. While recent advancements in end-to-end speech translation systems aim to eliminate intermediate source-language transcription during decoding, such methods often require extensive training data and complex architectures.

Typically, a speech translation (ST) system is composed of an automatic speech recognition (ASR) and a machine translation (MT) model, which is known as cascade system. However, in recent years, end-to-end models have gained popularity within the research community. These systems are encoder-decoder architectures capable of directly translating speech without intermediate symbolic representations. Nevertheless, while there are plenty of data available to train ASR and MT systems, there are not as many datasets for ST, despite some recent efforts [8]. Moreover, this approach is inherently more difficult because the encoder has to perform both acoustic modeling and semantic encoding. For these reasons, end-to-end ST systems still struggle to achieve the performance of cascade ST models.

Our system leverages a two-step approach for efficiency and modularity. We begin by processing Hindi audio input through an ASR model trained on OpenSLR, generating high-quality Hindi text transcriptions. Subsequently, the transcriptions are translated into English using a neural machine translation (NMT) model trained on Bilingual Corpus. Although the ASR and NMT models are trained independently, they are

*nandadebanjan.@gmail.com
†debayan.datta0206@gmail.com
‡ayanmaity813@gmail.com

seamlessly integrated during inference to perform an end-to-end pipeline that combines speech recognition and translation into a unified process.

This hybrid approach ensures flexibility and robustness while addressing practical constraints such as dataset availability and training scalability. By integrating two well-established systems, we achieve a reliable solution for Hindi-to-English speech translation, enabling effective communication across language barriers.

# 2 Related Works

The field of multilingual translation has gained significant attention for its ability to translate between multiple language pairs using a single model, thereby enhancing maintainability and improving performance on low-resource languages [12]. This concept has been extended to speech translation (ST) with one-to-many multilingual ST frameworks, such as adding language embeddings to each source feature vector [9]. These methods demonstrate that including the source language (e.g., English) as a target language boosts translation quality. A simpler approach was proposed by [11], where a target language token is prepended to the decoder, enabling one-to-many and many-to-many ST. However, many-to-one ST was not explored due to the lack of a large corpus, which was addressed by the release of the CoVoST dataset for ST from 11 languages into English [18]. This dataset has since enabled the development and evaluation of many-to-one ST systems.

The joint decoding of Automatic Speech Recognition (ASR) and ST was first proposed in a multi-task learning framework by [2]. Improvements in multitask ST have included using word embeddings as an intermediate representation [7]. Another notable approach involves a two-stage model performing ASR first and passing decoder states as input to a second ST model [2], blending cascaded translation with end-to-end trainability. To evaluate consistency between transcripts and translations, [16] introduced novel metrics and examined multiple model architectures. Their results showed that end-to-end models with coupled inference procedures achieve strong consistency. Additionally, they experimented with concatenating transcripts and translations as joint outputs for a shared encoder-decoder network. Although effective, this approach incurs higher latency compared to others, as the sequential nature of the outputs increases processing time.

The scarcity of task-specific data in ST systems has led to the adoption of pre-trained components. Pre-training ASR encoders has become a standard practice, as demonstrated in [5], achieving notable success in modern end-to-end systems [8]. However, pre-training the decoder for machine translation (MT) has shown limited benefits [4]. Pre-trained models like BERT are often utilized in Neural Machine Translation (NMT), either to initialize the NMT encoder [19, 13] or by feeding BERT's outputs into the NMT model as inputs [19, 15]. These approaches highlight the versatility of pre-trained language models in improving downstream tasks.

In speech recognition, advancements in deep learning have been pivotal. Traditional methods, such as Recurrent Neural Networks (RNNs) and convolutional networks, laid the foundation for modern speech recognition systems [14]. Two prominent methods are used for mapping variable-length audio sequences to transcriptions. The RNN encoder-decoder paradigm uses an encoder RNN to convert input into a fixed-length vector and a decoder to produce output sequences [6, 17]. The addition of attention mechanisms has significantly enhanced performance, particularly for longer sequences [3]. The second common approach employs the Connectionist Temporal Classification (CTC) loss function [10], coupled with RNNs to model temporal dependencies effectively.

These related works collectively provide a foundation for developing bilingual speech-to-text translation systems, such as the one presented in this project.

# 3 Proposed Methodology for Speech to Text Conversion

## 3.1 Data Preprocessing

Data preprocessing is a critical step in speech-to-text conversion to ensure the model receives input in a suitable and consistent format. The raw audio signals are first transformed into spectrograms using the Short-Time Fourier Transform (STFT). The STFT divides the audio signal into small overlapping sections or frames. For each frame, the Fourier Transform computes the frequency components, producing a frequency-time representation. The resulting spectrogram is a two-dimensional matrix where rows correspond to frequency bands and columns represent time intervals. The intensity of each point in this matrix indicates the amplitude of a specific frequency at a given time.

Spectrograms are particularly effective for speech data as they capture dynamic variations in frequency over time, essential for distinguishing phonemes and understanding linguistic content. However, the raw

spectrogram values may vary significantly across samples due to differences in audio recording conditions, volume levels, or noise. To address this, normalization is applied to scale the spectrogram values to a uniform range. This ensures consistency, improving the model's ability to generalize across varied data.

Additionally, preprocessing may involve noise reduction to eliminate background interference and increase clarity. Silence segments at the beginning and end of audio clips are often trimmed to focus the model's attention on meaningful speech. These steps collectively enhance the quality and interpretability of the input data for the subsequent stages.

## 3.2 Data Preparation

Preparing data for a speech-to-text system involves transforming both the audio and corresponding text transcriptions into formats suitable for model training. The input audio data is first converted into spectrograms during preprocessing, providing a structured representation of the sound's temporal and frequency characteristics. These spectrograms, with their high-dimensional features, form the input for the DeepSpeech2 model.

The output, or ground truth, consists of textual transcriptions of the audio clips. To make these textual data compatible with the model, transcriptions are tokenized into sequences of individual characters. Each character, including letters, spaces, and punctuation marks, is assigned a unique integer ID. This numeric representation allows the model to treat the transcription as a sequence of numbers, simplifying the processing of text data.

For instance, the text "hello" might be tokenized as [8, 5, 12, 12, 15] based on a predefined character-to-integer mapping. Padding or truncation is applied to ensure that all input and output sequences have consistent lengths during batch processing. The character IDs are later converted back to text during inference for readability and evaluation.

To ensure balanced data, the training set is curated to include diverse samples of speech patterns, accents, and audio qualities. Data augmentation techniques, such as adding noise or pitch shifts, may also be used to enhance the robustness of the model to real-world variations.

## 3.3 DeepSpeech2 Model Architecture

**Convolutional Layers:**
The spectrograms generated from audio files are fed into convolutional layers. These layers extract spatial features from the input by detecting patterns in frequency and time domains.[1] For example, a rising tone or a burst of energy at specific frequencies can be captured as features. Multiple convolutional layers are stacked, each followed by batch normalization to stabilize learning and activation functions (e.g., ReLU) to introduce non-linearity. This preprocessing step reduces noise and enhances the signal quality, providing compact feature maps for the next stage.

**Recurrent Layers:**
The feature maps are then passed to a stack of Bidirectional Gated Recurrent Units (Bi-GRUs). These layers excel at capturing temporal dependencies in sequential data. Unlike unidirectional layers, bidirectional layers process the sequence in both forward and backward directions. This allows the model to understand the context of a sound both before and after it occurs, which is crucial for speech recognition. For instance, identifying a vowel sound might depend on the preceding or succeeding consonant.

**Fully Connected Layers:**
After the recurrent layers, the processed features are flattened and passed through fully connected layers. These dense layers further abstract the features into a high-level representation. The output layer uses a softmax function to predict the probability distribution over all possible characters, including a special blank token for alignment.

**CTC Loss Function:**
The Connectionist Temporal Classification (CTC) loss function is a key component of DeepSpeech2. Traditional models require precise alignment between audio frames and characters, which is challenging to obtain. CTC eliminates this need by aligning predicted sequences with ground truth transcriptions at a sequence level. The blank token in the output allows flexibility in handling variable-length inputs and outputs, making the model robust to variations in speaking speed or pauses.

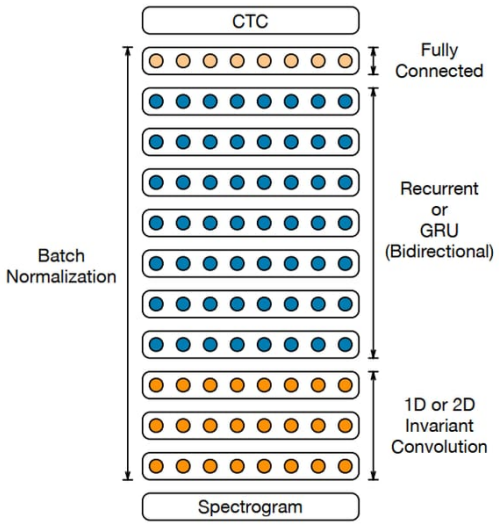


Figure 1: Architecture of DeepSpeech2

### 3.4 Workflow

The workflow starts by converting raw audio signals into spectrograms using STFT. These spectrograms provide a time-frequency representation of the audio, serving as input to the DeepSpeech2 model. The processing steps in the model include:

- **Audio Preprocessing:**
  Raw audio signals are first converted into spectrograms using the Short-Time Fourier Transform (STFT). This step provides a time-frequency representation of the sound, where patterns in frequency variations over time are captured. The spectrogram values are normalized to ensure consistency across different audio samples.

- **Feature Extraction:**
  The spectrograms are fed into the convolutional layers of the DeepSpeech2 model. These layers extract local spatial patterns, such as energy bursts or harmonic structures, and pass them to deeper layers for further processing.

- **Temporal Analysis:**
  The Bidirectional GRU layers analyze the sequential data to understand the relationships between different parts of the audio. These layers process the audio forward and backward in time, capturing both preceding and succeeding contexts, which is essential for accurate transcription.

- **Character Prediction:**
  The fully connected layers at the end of the network output probabilities for each character in the vocabulary. These probabilities are decoded into sequences of characters, forming the raw predictions.

- **Character Prediction:**
  The fully connected layers at the end of the network output probabilities for each character in the vocabulary. These probabilities are decoded into sequences of characters, forming the raw predictions.

- **Post-processing:**
  The predicted numeric sequences are converted back into readable text using the predefined character mapping. Post-processing may also involve removing extra spaces or correcting obvious errors.

### 3.5 Model Training and Evaluation

Model training involves optimizing the DeepSpeech2 architecture using a well-structured dataset. The spectrograms generated from audio files and their corresponding numeric transcriptions serve as input and output, respectively. The Adam optimizer is used to adjust the model's weights to minimize the CTC loss. Training is performed in batches, with each batch containing a mix of audio samples to ensure diversity.

The dataset is split into training and validation sets. The training set is used to update the model weights, while the validation set monitors the model's performance. Metrics like loss and Word Error Rate (WER) are tracked after each epoch to assess convergence. Data augmentation techniques, such as adding noise or varying pitch, may be applied to the training data to improve robustness.

During evaluation, the model's predictions are compared with the ground truth transcriptions using WER. Errors like insertions, deletions, and substitutions are quantified. Manual inspection of random samples helps identify specific issues, such as handling accents or noisy environments. Regular visualization of spectrograms and waveforms ensures the model is processing audio features effectively.

### 3.6 Evaluation Metrics

Word Error Rate (WER) is the primary metric for evaluating the model's performance. It quantifies the accuracy of transcriptions by comparing the predicted text to the ground truth. A lower WER indicates better transcription quality. Additionally, qualitative analysis is performed by manually reviewing random predictions and visualizing spectrograms alongside their transcriptions. This helps identify specific areas where the model might need improvement, such as handling accents or noisy audio.

## 4 Proposed Methodology for Machine Translation

### 4.1 Data Preprocessing

Data preprocessing is a critical step in preparing the Hindi-English dataset for training a translation model. The dataset consists of parallel sentences in both Hindi and English. Initially, rows with missing values are removed to ensure data consistency, as incomplete rows can disrupt training and evaluation processes. Next, the text in both languages is converted to lowercase to standardize the input, minimizing variations caused by case sensitivity. Punctuation marks, special characters, and numeric digits are then removed using Python's `re` module and string functions. This step ensures that the input text contains only meaningful words, simplifying the tokenization process.

To eliminate redundancy, the dataset is deduplicated by identifying and removing duplicate rows. Sentences longer than 20 words in either Hindi or English are filtered out. Long sequences can introduce noise and complicate training by requiring excessive computational resources. Additionally, English sentences are augmented with special start (`START_`) and end (`_END`) tokens. These tokens help the model recognize where each sentence begins and ends, facilitating better sequence alignment during training. After preprocessing, the dataset is clean, uniform, and ready for further preparation steps such as tokenization and embedding.

### 4.2 Data Preparation

Once preprocessing is complete, the dataset is prepared for training by splitting it into training and testing subsets. This step ensures that the model is evaluated on unseen data. The splitting process follows an 80-20 or 70-30 ratio to allocate sufficient data for both training and validation. Each unique word in the dataset is assigned an integer ID to create separate vocabularies for Hindi (input language) and English (output language). This mapping converts sentences into sequences of integers, which are essential for numerical processing by the model.

The sizes of the Hindi and English vocabularies are calculated, including an additional token for padding. Padding ensures that all sentences in a batch have the same length, enabling efficient matrix operations during training. The maximum sentence lengths in both languages are determined, and shorter sentences are padded with a special token (`<pad>`) to match these lengths. This standardization of input dimensions prevents dimension mismatch errors.

To enhance training efficiency, the dataset is shuffled to randomize the order of samples, mitigating any unintentional bias in the data. The integer mappings and vocabulary sizes are stored in dictionaries for quick access during encoding and decoding. This structured preparation lays the groundwork for creating batches and feeding data into the seq2seq model architecture.

### 4.3 Seq2Seq Model Architecture

The proposed Hindi-to-English translation task leverages a sequence-to-sequence (seq2seq) model with an encoder-decoder architecture. This design effectively captures temporal dependencies in sequential data, making it well-suited for translation tasks. The model is built using Long Short-Term Memory (LSTM) networks, which are adept at handling long-term dependencies by mitigating the vanishing gradient problem.

- Encoder: The encoder processes Hindi input sentences and generates a compact representation that captures the semantic essence of the sequence. It begins with an embedding layer that maps each integer-encoded Hindi word into a dense vector representation of fixed size. These embeddings serve as inputs to an LSTM layer, which processes the sequence one word at a time. The LSTM outputs two states: the hidden state and the cell state. These states encapsulate the sequence's context and are passed to the decoder for generating the translation.

- Decoder: The decoder generates English translations by leveraging the encoder's output states. It also starts with an embedding layer, which transforms integer-encoded English words into dense vector representations. These embeddings, along with the encoder's hidden and cell states, are fed into the decoder's LSTM layer. At each step, the LSTM produces a context-aware representation, which is passed through a dense layer with a softmax activation function. The dense layer computes a probability distribution over the target vocabulary, enabling the model to predict the next word in the sequence.

The model is trained using the categorical cross-entropy loss function and RMSprop optimizer. The architecture employs teacher forcing during training, where the decoder is provided the actual target sequence as input. The *generate_batch* function prepares data batches for the model by one-hot encoding the target sequences.

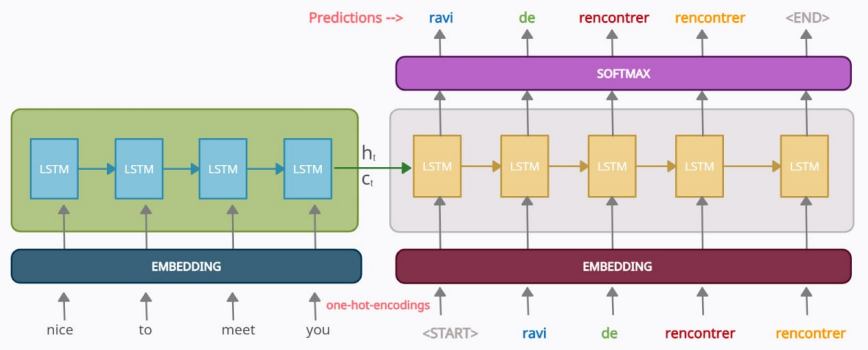


Figure 2: Seq2Seq Architecture

## 4.4 Workflow

The translation workflow begins with dataset preprocessing and preparation, which includes cleaning, tokenization, and splitting into training and test sets. Once the data is prepared, the training data is converted into padded sequences for both Hindi and English sentences. These sequences are fed into the seq2seq model for training.

The encoder processes the Hindi input and generates context vectors, summarizing the input sequence. These vectors are passed to the decoder, which uses them along with its internal states and the previously generated word to predict the next word in the output sequence. Teacher forcing ensures that the decoder learns efficiently by comparing predicted outputs to actual target words.

After training, the encoder and decoder are separated for inference. During inference, the decoder iteratively predicts one word at a time, using its previous output as input. This iterative prediction continues until the end token (`_END`) is generated. The `decode_sequence` function manages this process, allowing the model to translate unseen Hindi sentences.

## 4.5 Model Training and Evaluation

The model is trained on batches of data generated by the `generate_batch` function. Each batch consists of padded input-output pairs, which are processed by the encoder and decoder. The training objective is to minimize the categorical cross-entropy loss, which measures the difference between predicted and actual word distributions. The RMSprop optimizer is used to update the model's parameters iteratively.

During each epoch, the model's performance is evaluated on a validation set. Metrics such as training and validation loss are monitored to detect overfitting or underfitting. Early stopping is employed if the validation loss stagnates or increases, preventing unnecessary training.

Once training is complete, the model's weights are saved for future use. The validation set is used to qualitatively assess the model's translation accuracy. Predicted translations are compared to ground truth sentences, and discrepancies are analyzed to identify potential improvements.

### 4.6 Evaluation Metrics

The primary metric for evaluating the model is Word Error Rate (WER), which quantifies translation accuracy. WER measures the number of insertions, deletions, and substitutions required to match the predicted sequence with the actual sequence. Lower WER indicates better performance. Additionally, qualitative analysis of predicted translations helps validate alignment between input Hindi sentences and output English sequences.

## 5 Experimental Results:

### 5.1 Speech to Text:

The DeepSpeech2 model was trained for five epochs due to limitations in computational resources. Despite this restricted training duration, the model showed significant improvements in performance, as measured by the Word Error Rate (WER), which is a standard metric for evaluating speech-to-text systems. WER measures the number of errors in the predicted transcription compared to the actual transcription, with lower values indicating better accuracy.

The WER values for each epoch demonstrate steady progress as the model learned to better convert Hindi speech to text. The WER scores are summarized below:

| Epoch | WER |
|---|---|
| 1 | 1.0267 |
| 2 | 0.8205 |
| 3 | 0.6879 |
| 4 | 0.6118 |
| 5 | 0.6072 |

Table 1: WER results on validation data

These results show that the model adapted effectively to the task, with noticeable improvements after each epoch. In particular, the drop in WER between epochs 3 and 5 highlights the model's ability to refine its predictions and capture the nuances of the Hindi language.

While the results are promising, the training was limited due to resource constraints. With additional computational power and extended training time, the model could potentially achieve even better accuracy and lower WER values.

Here are some of the results from our machine Speech to Text task:

```
Target     : कहावकील तो सेठ जी हैं
Prediction: कहावतील तो से जी है

Target     : जैसे पिछले दो दिनों से मौनव्रत धारण कर रखा हो और
Prediction: कैसी किसले तोदिना सी मोलवरसता कर रका औ और

Target     : अगर मूंशी सत्यनाराण की नीयत खराब होती
Prediction: पर मूंशी सतेनाराण जीनय करा होती

Target     : मेरा पे कभी भी बर् से नहीं ढकेगा
Prediction: मेरा पे कभी भी बरब से नहीं बठेगा

Target     : हामिद का दिल बैठ गया
Prediction: हम मिद का दिर बेठ गया
```

In conclusion, the Hindi speech-to-text model demonstrated significant progress over the five epochs. This indicates that the approach is effective and has the potential for further improvement.Future work will focus on training the model for more epochs and using more advanced hardware to achieve state-of-the-art performance in Hindi speech-to-text conversion.

### 5.2 Machine Translation:

The Hindi-to-English text translation task was implemented using a sequence-to-sequence deep learning model based on an encoder-decoder architecture. This approach is widely used for natural language processing tasks such as translation, where the goal is to map a sequence in one language to a sequence in another. The model was trained over 100 epochs, with both the training loss and validation loss tracked to monitor its performance and generalization ability.

At the beginning of training, the model exhibited gradual improvements in both training and validation losses, indicating effective learning. The training loss decreased steadily from 6.14 in the first epoch to 3.28 by the 15th epoch, reflecting the model's increasing ability to predict accurate translations on the training data. Similarly, the validation loss reduced from 5.86 to 5.05 during the same period, showcasing its ability to generalize to unseen validation data. However, after epoch 15, the validation loss began to plateau and eventually increased slightly, signaling the onset of overfitting. By the 100th epoch, the training loss had dropped significantly to 0.34, while the validation loss increased to 7.11, confirming that the model had started to memorize the training data at the expense of generalization.

This training behavior highlights the need for additional techniques such as early stopping, dropout, or other regularization methods to mitigate overfitting and improve the model's performance on unseen data. Despite the overfitting, the model established a strong baseline for Hindi-to-English translation, achieving reasonable accuracy on training data and providing a foundation for further enhancements in both architecture and training strategies. This project demonstrates the potential of deep learning models in handling complex language translation tasks while underlining the importance of balancing training accuracy and generalization.

Here are some of the results from our machine translation task:

```
Input Hindi sentence: पिछले वर्ष विभिन्न कारणों से
Actual English Translation:  last year for various reasons
Predicted English Translation:  last year for various reasons

Input Hindi sentence: ऐसे व्यक्ति के उदाहरण के रूप में लिया
Actual English Translation:  as an example of a gentleman
Predicted English Translation:  as an example of a gentleman

Input Hindi sentence: इसका अर्थ है कि भीतर से आप जैसे हैं उसमें ख़ुशी पायें
Actual English Translation:  its just about being you and being cool with that
Predicted English Translation:  its just about the way you asked it got to eigh

Input Hindi sentence: यहाँ बच्चों ने अपना खुद का संगीत रिकॉर्ड किया
Actual English Translation:  where the children recorded their own music
Predicted English Translation:  where the children have taken cricket to go
```

| Data | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 |
|---|---|---|---|---|
| Train | 0.76146 | 0.74229 | 0.74479 | 0.75243 |
| Test | 0.187003 | 0.29371 | 0.38662 | 0.44772 |

Table 2: BLEU Scores for Train and Test Data

## 6 Future Work:

In the future, we plan to train our DeepSpeech2 model for more epochs to enhance its performance in speech-to-text conversion. By increasing the number of training epochs, we aim to fine-tune the model further, allowing it to produce more accurate and reliable transcription results. Additionally, we recognize the importance of addressing the overfitting issue in our machine translation system. By focusing on improving

its generalization capabilities, we hope to achieve better translation quality and make the system more robust. Once we have optimized the models and ensured they perform well, our next step will be to deploy them for real-world applications. To make these advancements more accessible and user-friendly, we are also working on developing a website. This platform will allow users to interact with both the speech-to-text and machine translation systems seamlessly, providing an intuitive and efficient way for people to benefit from these technologies in various practical scenarios.

# References


[1] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR, 2016.

[2] Antonios Anastasopoulos and David Chiang. Tied multitask learning for neural speech translation. *arXiv preprint arXiv:1802.06655*, 2018.

[3] Dzmitry Bahdanau. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. *arXiv preprint arXiv:1809.01431*, 2018.

[5] Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228. IEEE, 2018.

[6] Kyunghyun Cho. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Shun-Po Chuang, Tzu-Wei Sung, Alexander H Liu, and Hung-yi Lee. Worse wer, but better bleu? leveraging word embedding as intermediate in multitask end-to-end speech translation. *arXiv preprint arXiv:2005.10678*, 2020.

[8] Mattia A Di Gangi, Matteo Negri, Viet Nhat Nguyen, Amirhossein Tebbifakhr, and Marco Turchi. Data augmentation for end-to-end speech translation: Fbk@ iwslt '19. In *Proceedings of the 16th International Conference on Spoken Language Translation*, 2019.

[9] Mattia Antonino Di Gangi, Matteo Negri, and Marco Turchi. One-to-many multilingual end-to-end speech translation, 2019.

[10] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[11] Hirofumi Inaguma, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. Multilingual end-to-end speech translation. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 570–577. IEEE, 2019.

[12] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's multilingual neural machine translation system: Enabling zero-shot translation, 2017.

[13] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota, 2019.

[14] Tony Robinson, Mike Hochberg, and Steve Renals. The use of recurrent neural networks in continuous speech recognition. In *Automatic Speech and Speaker Recognition: Advanced Topics*, pages 233–258. Springer, 1996.

[15] Justyna Sarzynska-Wawer, Aleksander Wawer, Aleksandra Pawlak, Julia Szymanowska, Izabela Stefaniak, Michal Jarkiewicz, and Lukasz Okruszek. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research*, 304:114135, 2021.

[16] Matthias Sperber, Hendra Setiawan, Christian Gollan, Udhyakumar Nallasamy, and Matthias Paulik. Consistent transcription and translation of speech. *Transactions of the Association for Computational Linguistics*, 8:695–709, 2020.

[17] I Sutskever. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[18] Changhan Wang, Anne Wu, and Juan Pino. Covost 2 and massively multilingual speech-to-text translation. *arXiv preprint arXiv:2007.10310*, 2020.

[19] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. Incorporating bert into neural machine translation, 2020.