BiLSTM for ASR Automatic speech recognition

Daniele Cecca
Matr. 914358
MSc Artificial Intelligence for Science and Technology
Email: d.cecca@campus.unimib.it

1. Introduction

Automatic Speech Recognition (ASR) has emerged as a fundamental technology in natural language processing, enabling seamless interaction between humans and machines through spoken language. Applications such as voice assistants, transcription services, and accessibility tools increasingly rely on robust ASR systems capable of handling diverse acoustic conditions and linguistic variability. While recent advances in end-to-end neural architectures—such as Transformer-based models—have pushed state-of-the-art performance, recurrent neural networks (RNNs) with Connectionist Temporal Classification (CTC) loss remain a strong and widely adopted approach due to their efficiency, alignment-free training, and interpretability.

In this work, we present the design, training, and evaluation of an ASR system implemented using PyTorch and torchaudio. The model is based on a bidirectional Long Short-Term Memory (BiLSTM) network trained with the CTC loss function, which allows the system to directly map sequences of acoustic features to subword tokens without the need for pre-aligned transcripts. Audio data is preprocessed into Mel-frequency spectrograms, tokenized with SentencePiece to construct a subword vocabulary, and then passed through the BiLSTM-CTC architecture for transcription.

The system is evaluated using Word Error Rate (WER), a standard metric for ASR performance, with additional analysis provided through visualization of feature representations and output sequences. By demonstrating the end-to-end pipeline—from raw audio preprocessing to model inference and evaluation.

2. Data

For this work, we use the Common Voice dataset (version `cv-corpus-12.0-delta-2022-12-07`), released by the Mozilla Foundation. The Italian subset employed in our experiments contains 7,952 audio tracks, each paired with a corresponding transcription. Alongside the speech and text data, the corpus also provides additional metadata such as age, gender, accent, and voting information. However, for the purposes of this study, we rely exclusively on the audio signals and their transcriptions, as they constitute the core of the Automatic Speech Recognition (ASR) task.

2.1. Data Exploration

The dataset is stored in tabular format with 10 columns. Each entry includes a unique client identifier, the file path to the audio recording, the corresponding transcription, and additional metadata. An excerpt of the dataset structure is summarized below:

- Entries: 7,952
- Columns: client_id, path, sentence, up_votes, down_votes, age, gender, accents, locale, segment

The audio files are provided in MP3 format with a sampling rate of 32 kHz, mono channel, and encoded using MPEG Layer III compression. A representative example is the file `common_voice_it_36409516.mp3`, which has a duration of 4.04 seconds.

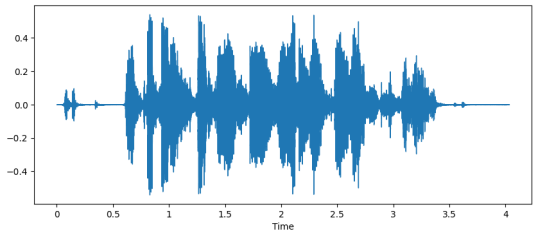

Figure 1: Audio sample

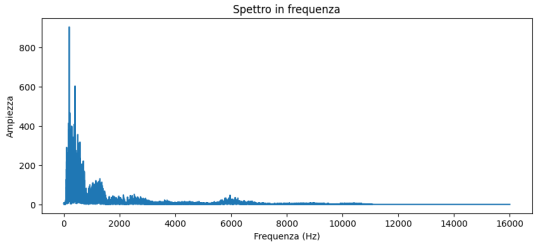

Figure 2: Spectrum sample

2.2. Sampling and Quantization

To ensure uniformity across the dataset, all audio tracks are resampled to 16 kHz and converted to mono channel. This sampling rate is a widely adopted standard for ASR,

as it preserves the frequency range most relevant to human speech while reducing computational costs.

The audio is also re-encoded from MP3 to WAV format with 16-bit PCM quantization, providing a lossless representation. This step eliminates compression artifacts inherent to MP3 files and ensures consistent, high-quality input for subsequent feature extraction and model training.

2.3. Windowing and Mel-Feature Extraction

For each audio file, Mel-spectrogram features are extracted as a perceptually motivated representation of speech. The process involves short-time Fourier analysis with overlapping windows, followed by mapping the resulting frequency bins onto the Mel scale.

The Mel scale is designed to approximate the human auditory system, emphasizing low-frequency differences (e.g., 300 Hz vs. 400 Hz) and compressing high-frequency regions (e.g., 8,000 Hz vs. 8,100 Hz). The resulting Mel-spectrograms thus provide a more effective representation for speech recognition tasks than raw waveforms or linear spectrograms.

In this case, I adopted a window size of 400 samples (corresponding to approximately 25 ms at 16 kHz) with a hop size of 160 samples (approximately 10 ms). Furthermore, the Mel filterbank was applied over the frequency range of 20–7,600 Hz, using 80 Mel bands to obtain the final feature representation.

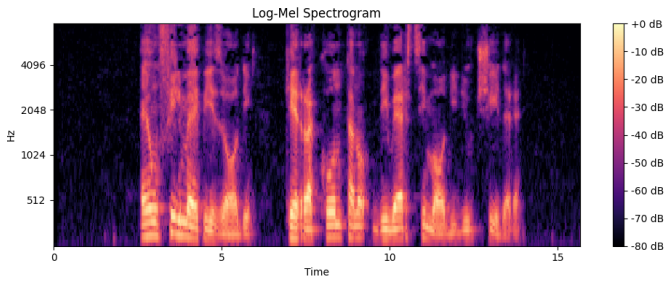

Figure 3: Mel-spectrograms

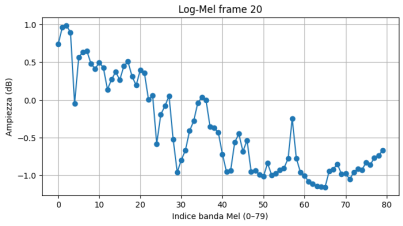

Figure 4: One log-mel frame

2.4. Preprocessing: Normalization

The transcription text is normalized through a series of preprocessing steps. All characters are first converted to lowercase, and leading or trailing whitespace is removed. Non-desired characters, such as punctuation, are filtered out, while accented characters in Italian (e.g., à, è, ì, ò, ù) are preserved. Finally, multiple consecutive spaces are collapsed into a single space.

For example, the transcription *"Leo Wilden!!"* is normalized to *"leo wilden"*. This step reduces variability in the dataset and improves the robustness of the tokenization and training stages.

2.5. Tokenization

To map transcriptions into subword units, SentencePiece tokenization with the unigram language model is employed. SentencePiece constructs a vocabulary in a data-driven manner, enabling the system to effectively handle out-of-vocabulary words while maintaining compact representation.

The vocabulary size is set to 1,000 subword units, with an additional token for whitespace, yielding a final vocabulary size of 1,001. For example, the tokenized sequence for the normalized transcription *"leo wilden"* is:

- Token IDs: [46, 5, 243, 7, 37, 62, 10]
- Decoded sequence: *"leo wilden"*

This tokenization approach ensures a flexible balance between character-level and word-level modeling, which is particularly effective in ASR tasks where rare words and diverse linguistic forms frequently occur.

3. Model

3.1. Class Dataset

To feed the network, a custom `Dataset` class was implemented. The input features consist of Mel-spectrogram representations of the audio signals, while the target sequences are the tokenized transcriptions. Inside the constructor, the text is encoded into subword units suitable for CTC training. Each element of the dataset therefore returns a tuple of Mel features and the corresponding integer-encoded transcription sequence.

The dataset was split into training, validation, and test sets as follows:

- Training set: 7,156 utterances
- Validation set: 796 utterances
- Test set: 1,591 utterances

Data were loaded using `DataLoader` with a batch size of 8 for training and validation, and 1 for testing. A custom collation function was required to properly handle variable-length sequences. The function `collate_ctc` takes a batch of samples, each represented as a pair (mel, ids), where $\text{mel} \in \mathbb{R}^{80\times T}$ is the Mel-spectrogram of variable time length T, and $\text{ids} \in \mathbb{N}^{L}$ is the corresponding sequence of token IDs with length L.

The collation process performs the following steps:

1) Compute the maximum input length $T_{\max}$ across the batch.

2) Initialize a padded tensor of size $(B, 80, T_{\max})$ filled with zeros, where B is the batch size.
3) Copy each spectrogram into the corresponding slice of this tensor, leaving the remaining positions as padding.
4) Concatenate all target sequences into a single one-dimensional tensor, as required by the PyTorch CTC loss function.
5) Record two auxiliary vectors:
   - `input_lengths` of shape $(B,)$, storing the true length of each input sequence before padding.
   - `target_lengths` of shape $(B,)$, storing the length of each target transcription.

This procedure ensures that batches with variable-length utterances can be processed efficiently.

Without such a collation step, batching variable-length speech inputs would not be feasible, as both the audio features and their corresponding transcriptions vary in length.

3.2. BiLSTMCTC

The network designed for Automatic Speech Recognition consists of three main building blocks:

1) Prenet (Feature computation): applies layer normalization and a linear projection to map Mel features into a hidden representation, followed by a ReLU activation.
2) Encoder (BiLSTM): a stack of three bidirectional LSTM layers with 256 hidden units, which captures temporal dependencies in both forward and backward directions.
3) Classifier: a linear layer mapping the concatenated forward and backward LSTM outputs to the vocabulary logits (1,001 tokens).

The complete architecture can be summarized as:

```
BiLSTMCTC(
  (prenet): Sequential(
    (0): LayerNorm((80,), eps=1e-05,
        elementwise_affine=True)
    (1): Linear(in_features=80, out_features=256,
        bias=True)
    (2): ReLU()
  )
  (lstm): LSTM(256, 256, num_layers=3,
      batch_first=True, bidirectional=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=512,
      out_features=1001, bias=True)
)
```

3.3. Training

The model was trained for 50 epochs using the Connectionist Temporal Classification (CTC) loss, which is specifically designed for sequence-to-sequence problems such as speech recognition where input and output lengths differ and alignments are unknown. The intuition of CTC is to output a single character for every frame of the input, so that the output is the same length as the input, and then to apply a collapsing function that combines sequences of identical letters, resulting in a shorter sequence. CTC introduces a "blank" symbol that can be used in the alignment whenever we don't want to transcribe a letter.

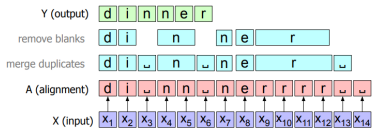

Figure 5: CTC collapsing function

During training, the model with the lowest validation loss was saved to avoid overfitting. Figure 6 shows the training and validation loss curves across the 50 epochs. The training loss steadily decreased, while the validation loss initially decreased but started to rise slightly after epoch 20, indicating some degree of overfitting.

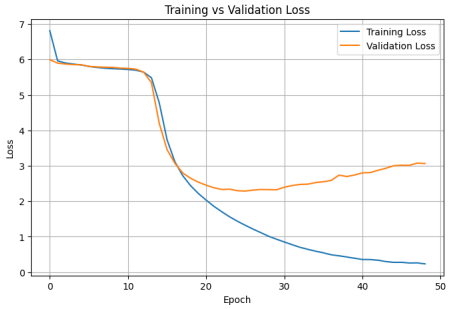

Figure 6: Training

3.4. Testing

The performance of the trained model was evaluated on the test set using the Word Error Rate (WER) metric. To get the output utterance I used a greedy approach; CTC makes a strong conditional independence assumption: it assumes that, given the input X, the CTC model output at time t is independent of the output labels at any other time a_i. Thus:

$$P_{\text{CTC}}(A \mid X) = \prod_{t=1}^{T} p(a_t \mid X) \tag{1}$$

To find the best alignment $\hat{A} = \{\hat{a}_1, \ldots, \hat{a}_T\}$, we can greedily choose the character with the maximum probability at each time step t:

$$\hat{a}_t = \arg\max_{c \in \mathcal{C}} \; p_t(c \mid X) \tag{2}$$

WER measures the difference between the predicted transcription and the ground truth by computing the minimum number of substitutions (S), deletions (D), and insertions (I)

required to transform one into the other, normalized by the total number of words (N) in the reference:

$$WER = \frac{S + D + I}{N}$$

On the 1,591 test utterances, the model achieved a WER of 0.633. An example of prediction versus ground truth is shown below:

- Reference: “è infiammabile e nocivo per inalazione e contatto con la pelle”
- Prediction: “è inmm mabile è una civo ver inlazione e contatto con la bell”

Although the model was able to capture the general structure of the sentence, several substitution and insertion errors remain, reflecting the challenge of training ASR systems on relatively limited datasets.

4. Conclusion

In this work, we presented an Automatic Speech Recognition (ASR) system based on a BiLSTM network trained with Connectionist Temporal Classification (CTC) loss. The system was developed using PyTorch and torchaudio, and it included all key stages of an end-to-end ASR pipeline: preprocessing, feature extraction, tokenization, model training, and evaluation. The model achieved a Word Error Rate (WER) of 0.633 on the test set, demonstrating that even relatively simple architectures can learn meaningful mappings from audio features to text.

The results also highlighted the limitations imposed by the size of the training corpus. With only 7,952 utterances, the model showed signs of overfitting after around 20 epochs and struggled to generalize perfectly to unseen data. Nevertheless, the observed performance suggests that the proposed architecture is capable of learning robust acoustic and linguistic patterns.

Future improvements will primarily depend on the availability of larger and more diverse training datasets. By training on significantly more speech data, the model would be expected to achieve substantially better results, reducing the Word Error Rate and producing more accurate transcriptions.