Brain-to-Text '25

台灣科技大學 M11307509 康智惟

1. Introduction

This report details the implementation and methodology for a neural decoding solution developed for the Brain-to-Text '25 competition. The objective of the competition is to decode variable-length time series of neural activity, specifically from the speech motor cortex of participant 'T15', into transcribed English text. The proposed solution employs a robust, end-to-end deep learning architecture combining Convolutional Neural Networks (CNNs) and a Bidirectional Long Short-Term Memory (BiLSTM) network, optimized using the Connectionist Temporal Classification (CTC) loss function. This architecture, referred to as the Baseline CTC Model in the provided code, establishes a strong neural baseline against the competition's published benchmark of 6.70% Word Error Rate (WER).

2. Dataset and Preprocessing

The competition dataset comprises 10,948 spoken sentences across 45 sessions from a single participant ('T15') over 20 months. The neural input consists of 512 features (spike threshold crossings and spike band power from 256 electrodes) binned at 20 ms resolution.

2.1. Data Loading and Splitting

The data loading utility (load_split) is designed to handle the hierarchical HDF5 file structure, extracting neural features, actual time steps (n_time_steps), and corresponding sentence labels for the predefined train, val, and test partitions. For the final submission, test data is loaded with chronological tracking of session, block, and trial number to ensure correct CSV formatting.

2.2. Feature Normalization

Input neural features are standardized to mitigate variations across sessions and trials. Within the BrainToTextDataset and load_test_data_for_submission functions, Z-score normalization is applied to each feature sequence:

$$\text{Features}_{\text{norm}} = \frac{\text{Features} - \mu_{\text{features}}}{\sigma_{\text{features}} + 10^{-8}}$$

Additionally, features are clipped between ± 5 to prevent extreme outliers from destabilizing the training process.

2.3. Vocabulary and Tokenization

The model uses a character-level vocabulary built dynamically from the unique, lowercase characters present in the training sentences. Crucially, the vocabulary includes the essential CTC blank symbol, which is fixed at index 0. The target sentence is converted into a sequence of integer indices for training.

2.4. Batching and Padding

The collate_fn ensures efficient training by:

1. Sorting all sequences within a batch by length in descending order, a requirement for PyTorch's pack_padded_sequence.
2. Padding both the neural features and the target labels to the maximum length within the batch.

3. Model Architecture: Baseline CTC Model

The BaselineCTCModel is an acoustic-style end-to-end sequence prediction network designed for speech tasks, adapted here for neural signals.

3.1. Architecture Summary

Layer Block	Type	Input Dimensions	Output Dimensions	Purpose
CNN Feature Extractor	Two 1D-Conv Blocks	$(B, T, 512)$	$(B, T, 256)$	Temporal context modeling and dimension reduction.

Layer Block	Type	Input Dimensions	Output Dimensions	Purpose
Recurrent Layer	3-Layer BiLSTM	$(B, T, 256)$	$(B, T, 2 \times 512)$	Captures long-range dependencies in both forward and backward temporal directions.
Output Layer	Linear (Fully Connected)	$(B, T, 1024)$	(B, T, V)	Projects LSTM output to the vocabulary size V, followed by Log Softmax.

3.2. Training Configuration

The model is trained using the following parameters:

- **Input Dimension:** 512 (Neural features).
- **Hidden Dimension:** 512.
- **Number of Layers:** 3.
- **Dropout:** 0.3 (Applied in CNN and between LSTM layers).
- **Loss Function:** CTCLoss(blank $= 0$,zero_infinity $=$ True).
- **Optimizer:** AdamW.
- **Learning Rate Schedule:** OneCycleLR $with\ a\ maximum$ lr $= 10^{-3}$.
- **Regularization:** Gradient clipping with a maximum norm of 5.0 is applied during training.

4. Evaluation and Decoding

4.1. Word Error Rate (WER)

The primary evaluation metric is the Word Error Rate (WER), calculated using the jiwer library. WER is defined as:

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

where S, I, and D are the number of substitutions, insertions, and deletions, and N is the total number of words in the true transcription.

4.2. Greedy Decoding

In the validation and prediction phases, the output log probabilities from the model, log_probs $\in R^{T \times B \times V}$, are decoded greedily. This process involves:

1. Taking the argmax across the vocabulary dimension V at every time step T.
2. Applying the CTC collapsing operation: removing all consecutive duplicate tokens and all blank tokens (index 0).
3. Mapping the resulting indices back to their corresponding characters.

5. Submission Pipeline

The pipeline is designed to generate the final submission.csv file with the required format.

1. **Test Data Preparation:** The load_test_data_for_submission function loads all test trials, applying normalization and critically preserving the chronological order as determined by session date, block, and trial number. This chronological order defines the final submission id (0 to 1449).
2. **Model Loading:** The best-performing model (lowest validation WER) is loaded from the best_model.pt checkpoint.
3. **Prediction Generation:** The generate_predictions function processes the chronological test samples in batches, performs model inference, and applies the greedy decoding strategy, ensuring the predicted text list aligns with the chronological order of the input samples.
4. **CSV Creation:** The create_submission_file function finalizes the submission, generating a CSV with two columns: id (0 to N-1) and text (the decoded sentence). Punctuation marks are automatically excluded as they are not included in the character vocabulary.

6. Conclusion and Future Directions

The implemented CNN-BiLSTM-CTC pipeline provides a solid, end-to-end approach to the Brain-to-Text '25 challenge. The architecture is structurally sound for sequence modeling and has been shown to achieve competitive performance against the established baseline.

Potential avenues for further improvement, as suggested in the competition brief, include:

- **Advanced Decoding:** Replacing greedy decoding with beam search to leverage probability distributions more effectively.
- **Language Models:** Integrating an external character or word-level Neural Language Model (NLM) into the decoding process for more coherent and contextually appropriate transcriptions.
- **Architecture Enhancements:** Exploring Transformer-based models or sequence-to-sequence architectures (e.g., RNN-T) for superior long-range dependency capture.