CS 753: Automatic Speech Recognition
Assignment #1 (35 points)

Instructor: Preethi Jyothi*

TAs: Dhiraj and Sona for Parts I and IV
Darshan, Bhavani and Ashish for Parts II(A) and II(B)
Vaibhav, Amruta and Sameep for Part III

Jan 29, 2024

Instructions: This assignment is due on or before **11.59 pm on February 18, 2024**. The submission portal on Moodle will be closed after midnight on February 18.

- This is a group assignment. One submission can be made per team.
- For this assignment, you will run all your code via Colab notebooks. Submissions will also be `ipynb` notebooks. **Important:** While submitting, please make sure that your submissions are fully run notebooks; do not clear the outputs. This will make it easier to cross-check what the TAs see at runtime with what you got.
- Submit a tgz file named `assgmt1.tgz` on Moodle. It should contain a `README.txt` that contains the names and roll numbers of all the team members. Any notes that you would like to convey to your TAs can be added to `README.txt`. Maintain the following directory structure within `assgmt1.tgz`:

```
+- README.txt
+- assgmt1-Q1.ipynb
+- assgmt1-Q2A.ipynb
+- assgmt1-Q2B.ipynb
+- assgmt1-Q3.ipynb
+- assgmt1-Q4.ipynb
```

  It is very important that you do not deviate from the specified structure. Deviations will be penalized.

Starting Material: For this assignment, you will be working with the SpeechBrain toolkit and a CTC-based Conformer model as the starting point. Click here for some resources to help you navigate through the assignment, especially if you are new to Transformers and/or PyTorch.

Part I: Running a SpeechBrain ASR Recipe [4 points]

This part will help familiarize you with the basic structure of a SpeechBrain ASR recipe. On completing this part, you will have run an end-to-end pure CTC-based ASR model trained from scratch on the mini Librispeech ASR corpus.

*Special acknowledgement to my ex-student Darshan Prabhu (class of 2023) who helped ideate and set up significant parts of this assignment.

Go to https://colab.research.google.com. Then, go to "File → Upload notebook" and upload the ipynb file at https://cse.iitb.ac.in/~pjyothi/cs753/assgmt1-Q1.ipynb. This recipe is currently incomplete. To complete the recipe, there are four minor tasks marked with "####TASK" in the code that you will need to fill in. Once it is complete, the recipe will take roughly 40 mins to complete training for 30 epochs. Your final validation CER and WER will be 38.59% and 77.61%, respectively, and test CER and WER will be 39.61% and 78.63%, respectively. (There might be minor differences in these numbers on account of the use of different Colab machines, which can be discounted.)

What to submit: You need to submit `assgmt1-Q1.ipynb`. We will run this cell-by-cell to produce the final test CER and WER. You will receive full 4 points if the four tasks are correctly resolved and your test CER/WER is statistically comparable to the expected error rates listed earlier in this question.

Useful reference: Refer to the SpeechBrain notebooks linked in our helper doc to get familiar with the structure of SpeechBrain ASR recipes.

Part II: Modifications to the Base CTC Conformer Model [15 points]

(A) CTC is all you need [5 points]

Connectionist Temporal Classification (CTC) loss is commonly used in end-to-end ASR systems. Given an ASR encoder, states from the last encoder layer are used to compute logits and further softmax-normalized to yield probabilities which are used in the CTC loss computation.

In this part, you will implement a variant that we dub as "**inter-CTC**". Instead of computing the CTC loss using only the last encoder layer, we can compute a sum of CTC losses across each layer within a set of intermediate encoder layers. For inter-CTC, we define two hyperparameters:

- `intermediate_layers` contains the indices of intermediate layers for which we want to impose a CTC loss (e.g., '2,4'). Note that these values are one-indexed.
- `interctc_weight` is a scaling factor used to scale the inter-CTC loss.

If $\mathcal{L}_{\text{ctc}}$ is the standard CTC loss computed using the final encoder layer, $\mathcal{L}_{\text{i-ctc}}$ is the inter-CTC loss summed over all the intermediate layers defined in `intermediate_layers` and `interctc_weight` is a value α, then the combined loss used to compute gradients would be $(1-\alpha)\mathcal{L}_{\text{ctc}} + \alpha\mathcal{L}_{\text{i-ctc}}$.

Download the code template for this part at https://cse.iitb.ac.in/~pjyothi/cs753/assgmt1-Q2.ipynb. Look at the code under "Part II(A)". Do not change the hyperparameter values for `intermediate_layers` and `interctc_weight` given in the code template. Look for TODOs in the code template; complete all these TODOs to successfully implement inter-CTC. In this part, you will learn about a handy tool called **PyTorch Hooks**. See here for an accessible blog about PyTorch hooks. You will specifically have use for a forward hook.

What to submit: You need to submit `assgmt1-Q2A.ipynb` with all the TODOS completed. You will receive full points for this question if the code is all correctly populated and your test CER and WER with inter-CTC is roughly 1 point lower than what you got with the base CTC model in Part I. (For your reference, we get a test CER of 38.47% and test WER of 77.52% using inter-CTC with the hyperparameter values specified in the code template.)

(B) The PowerConv Module [10 points]

The Conformer model comprises a convolution module sandwiched between two feedforward layers (and a multiheaded self-attention layer) with residual connections. In this part, we will replace the convolution module with a more powerful alternative that we call the "**PowerConv**" module. Figure 1 illustrates the architecture of a PowerConv block.

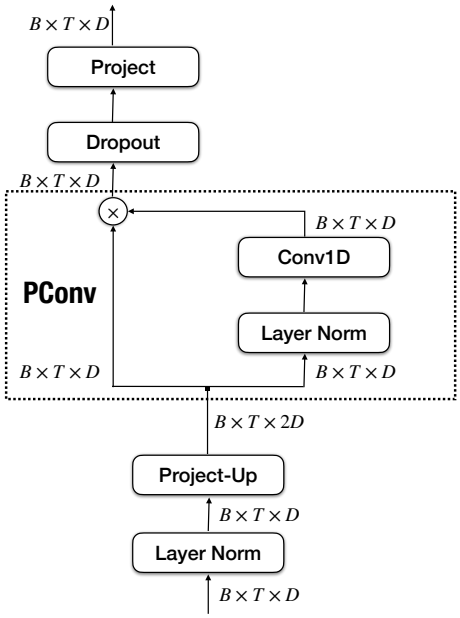

Figure 1: Architecture of the PowerConv Module

Consider an input tensor $\mathbf{X} \in \mathbb{R}^{B \times T \times D}$ with batch-size B, sequence length T and dimensionality D. The PowerConv module operates on $\mathbf{X}$ as follows:

$$\mathbf{X} \leftarrow \text{LayerNorm}(\mathbf{X}) \in \mathbb{R}^{B \times T \times D}$$
$$\mathbf{V} \leftarrow \mathbf{X}\mathbf{U} \in \mathbb{R}^{B \times T \times 2D} \quad \text{Comment: } \mathbf{U} \text{ is a first projection matrix}$$
$$\mathbf{Z} \leftarrow \mathbf{PConv}(\mathbf{V}) \in \mathbb{R}^{B \times T \times D} \quad \text{Comment: } \mathbf{PConv} \text{ is the main block shown in Figure 1}$$
$$\mathbf{O} \leftarrow \text{Dropout}(\mathbf{Z})\mathbf{W} \in \mathbb{R}^{B \times T \times D} \quad \text{Comment: } \mathbf{W} \text{ is a second projection matrix}$$

The **PConv** block first splits the input $\mathbf{V}$ equally along the feature dimension into $\mathbf{V}_1, \mathbf{V}_2 \in \mathbb{R}^{B \times T \times D}$. $\mathbf{V}_2$ goes through layer normalization followed by a depthwise 1D convolution along time (so that the output shape remains the same) to get $\mathbf{V}_2'$. The output of the **PConv** block, $\mathbf{Z}$, is computed as the element-wise product $\mathbf{V}_1 \otimes \mathbf{V}_2'$.

Download the code template for this part at https://cse.iitb.ac.in/~pjyothi/cs753/assgmt1-Q2.ipynb. Look at the code under "Part II(B)". Look for TODOs in the code template and complete all of them to successfully implement the Conformer encoder with the PowerConv module.

What to submit: You need to submit assgmt1-Q2B.ipynb with all the TODOS completed. You will receive full points for this question if the code is all correctly populated and your test WER with inter-CTC is roughly 3 points lower than what you got with the base CTC model in Part I. (For your reference, we get a test CER of 36.59% and test WER of 74.66% using the PowerConv module along with inter-CTC.)

Part III: Visualizing CTC [11 points]

CTC computation relies on frame-level probabilities. Hence, finding the best path within a CTC trellis gives an **alignment** between the audio and the tokenized string. In this part, you will implement the CTC forward algorithm from scratch and plot the CTC alignment for a specific test audio.

Download the code template for this part at https://cse.iitb.ac.in/~pjyothi/cs753/assgmt1-Q3.ipynb. Look at the code under "Part III". Look for TODOs in the code template and complete all six of them to successfully plot a CTC alignment like the one shown below in Figure 2.

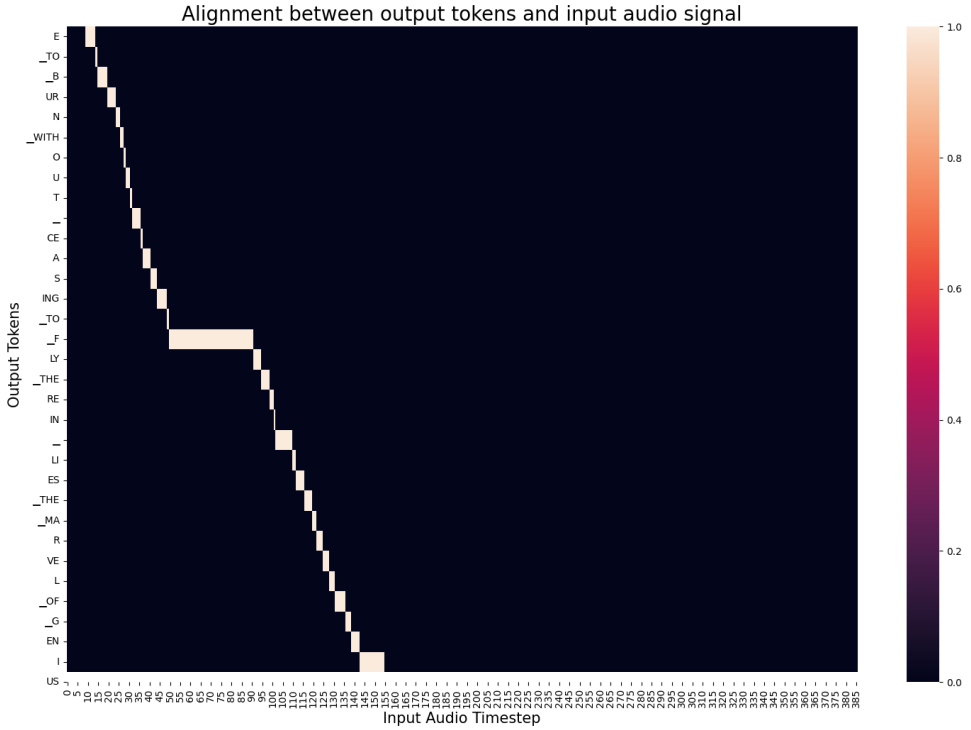

Figure 2: CTC alignment of a test audio with its transcription

What to submit: You need to submit assgmt1-Q3.ipynb with all six TODOS completed. You will receive full points for this question if the code template is correctly populated and we get the desired alignments for new test audio files that we will evaluate during grading.

Part IV: Climb the Leaderboard [5 points]

For this part, you have free reign to start from the Conformer model in Parts I or II and add any further modifications you like. Note that you cannot use more training data than what was used in Parts I/II. You will output predictions for a blind test set in https://cse.iitb.ac.in/~pjyothi/cs753/blindtest.tgz. Upload your predictions to the Kaggle task for this assignment to see where you appear on a leaderboard.

What to submit: If you attempt this part, you need to submit assgmt1-Q4.ipynb with your improved ASR system. You will get full five points if your test WER on the blind test set is lower than the test WER we get with the Conformer model in Part II(B), and we are able to successfully run your notebook. Further extra credit points will be given to teams on the top of the leaderboard.