Report on Training the Wav2Vec2 Model Using Common Voice Data

Outcome

The wav2vec2-large-960h model was trained in Google Colab using the L4 configuration, utilizing a GPU with 22.5GB of memory. Due to constraints in time and computational resources, the final model was trained on a subset of 100,000 samples (approximately 50% of the total data) for 10,000 steps with a batch size of 16, over a period of 7 hours. The resulting word error rates (WER) were 0.08 for the cv-valid-dev set and 0.077 for the cv-valid-test set. This represents a significant improvement, as the WER for the cv-valid-dev set was 0.108 prior to finetuning, resulting in a 26% reduction in WER following the finetuning process.

Observations

The training log indicates that all metrics, including training loss, validation loss, and WER, are still gradually decreasing at the 10,000 training step mark. This suggests that the model weights have not yet fully converged. Therefore, it is recommended that training continue beyond 10,000 steps until the validation loss and WER stabilize.

Proposed Next Steps

1. **Extended Training**: Continue training for a longer duration and utilize the most recent and largest Common Voice dataset to enhance model performance.
2. **Hyperparameter Optimization**: Since this was my first experience training a Wav2Vec2 model, I initially relied on guidelines from the Wav2Vec2 paper and similar training recipes, such as XLSR-Wav2Vec2, to establish effective hyperparameters. For the next training iteration, I plan to implement a learning rate scheduler, specifically the Cosine Annealing Scheduler, which is commonly used in training large language models (LLMs). Additionally, I mistakenly set the `return_attention_mask` parameter to False in the Wav2Vec2FeatureExtractor function; I will correct this to True in the upcoming trial.
3. **Data Augmentation**: Given that Common Voice data can be quite noisy, applying data augmentation techniques will be beneficial. This may include adding random noise, applying pitch shifts, or performing time-stretching on the speech signals. I plan to utilize the MLTU (Machine Learning Training Utilities) library, which conveniently provides functions for these augmentation techniques.

By implementing these next steps, I aim to further improve the performance of the Wav2Vec2 model and enhance its robustness against the challenges posed by noisy data.