Task 4: Training Report

Evaluation Methodology

For this task, both the original wav2vec2 model (from Task 2a) and the fine-tuned model (from Task 3) were evaluated on the cv-valid-dev dataset. The evaluation used Word Error Rate (WER) as the primary metric, where accuracy is calculated as:

$$\text{Accuracy} = 1 - \text{WER}$$

This metric was computed for each audio sample transcription, providing a quantitative comparison of the models' performance.

Results & Observations

- **Original Model**: Average accuracy = **0.883**
- **Fine-tuned Model**: Average accuracy = **0.883**

While the average accuracy for both models is identical, a deeper inspection of the transcriptions reveals differences in performance across individual samples. The fine-tuned model demonstrated improvement on certain clips. However, it also underperformed on other samples, compared to the original model.

Explanation: Overfitting Due to Small Training Set

I hypothesize that this discrepancy can be attributed to overfitting. Due to time constraints, I was limited to training the fine-tuned model on a relatively small subset of the full dataset. Specifically, I trained the model using only 350 training samples over 2 epochs.

Small Training Set (350 samples)

Fine-tuning a large, pre-trained model like wav2vec2 with a limited amount of data (350 samples) can lead to overfitting. The model may become too specialized in the patterns found in the small training set, leading to memorization of specific audio characteristics. As a result, the model performs well on the training set but struggles to generalize to new, unseen examples—like the samples from the cv-valid-dev dataset.

Small Validation Set (150 samples)

A limited validation set (150 samples) compounds the issue. If the validation data lacks sufficient diversity in terms of speech patterns, accents, and background noise, the model can be misled into thinking it has learned all the necessary features. The

validation set may not represent the full variability of real-world data, making it harder for the model to generalize beyond the narrow patterns it has seen.

Proposed Solutions to Improve Accuracy

To further improve the accuracy of the fine-tuned model, the following steps are recommended:

1. Increase Training & Validation Data Size:

Train the model on a larger dataset. Ideally, use the full cv-train-dev dataset or increase the training set with additional data from other sources, such as: LibriSpeech, TED-LIUM, VoxPopuli

Ensure that the validation set is larger and more representative of the target distribution. It should cover a variety of speakers, accents, and environmental conditions to provide a better estimate of model performance in real-world scenarios.

2. Data Augmentation:

Implement data augmentation techniques to artificially expand the training set, including:

- **Noise addition** (background noise, reverberation)
- **Pitch and speed variations**
- **Volume shifts** These transformations would force the model to learn more robust features, helping it perform better on unseen data.

3. Training Hyperparameters:

Standardize preprocessing across both training and evaluation data:

- **Batch sizes** currently due to computational limitations I am running a batch size of 2. A smaller batch size can introduce regularization effects due to noisy updates, which might help in low-data scenarios like mine. A larger batch size might stabilises the gradient updates but may require more data to avoid overfitting.
- **Number of Epochs,** More epochs allow the model to learn more, but with small datasets, this often leads to overfitting

4. Hyperparameter Search: Using Random Search:

Random search is better than a grid search in most cases. Instead of trying every possible combination like grid search, random search samples random combinations. It is more efficient as shown from studies like Bergstra & Bengio, 2012

Use random search to explore different combinations of hyperparameters. This involves defining a set of possible values for each hyperparameter and randomly

sample combinations of these values. Train and evaluate a model for each sampled combination. For example, a random search might explore variations of:

- **Learning rate**: [5e-5, 3e-5, 1e-5]
- **Batch size**: [8, 16, 32]
- **Dropout rate**: [0.1, 0.2, 0.3]
- **Number of epochs**: [2, 3, 4]

Benefits:

- Helps identify optimal training settings, especially when model performance is sensitive to changes.
- Can reveal interactions between parameters (e.g., a higher learning rate working well only with a smaller batch size).
- Provides empirical evidence for hyperparameter selection, rather than relying on defaults or trial-and-error.

6. Evaluate with Additional Metrics:

- Supplement accuracy and WER with other evaluation metrics:
  - **Character Error Rate (CER)** for finer-grained performance analysis.
  - **Semantic similarity metrics** like BLEU or ROUGE to evaluate how well the model captures meaning in transcriptions.