Training Report

1. Training Approach

This section details our approach on the training process and how certain decision were made.

1.1 Dataset

We will be using the cv-valid-train data, which comprises of ~200,000 audio mp3 files. Due to the limited amount of time and compute we have; we will train on a subset of 10,000 files (5% of original training data).

1.2 Model Training Parameters

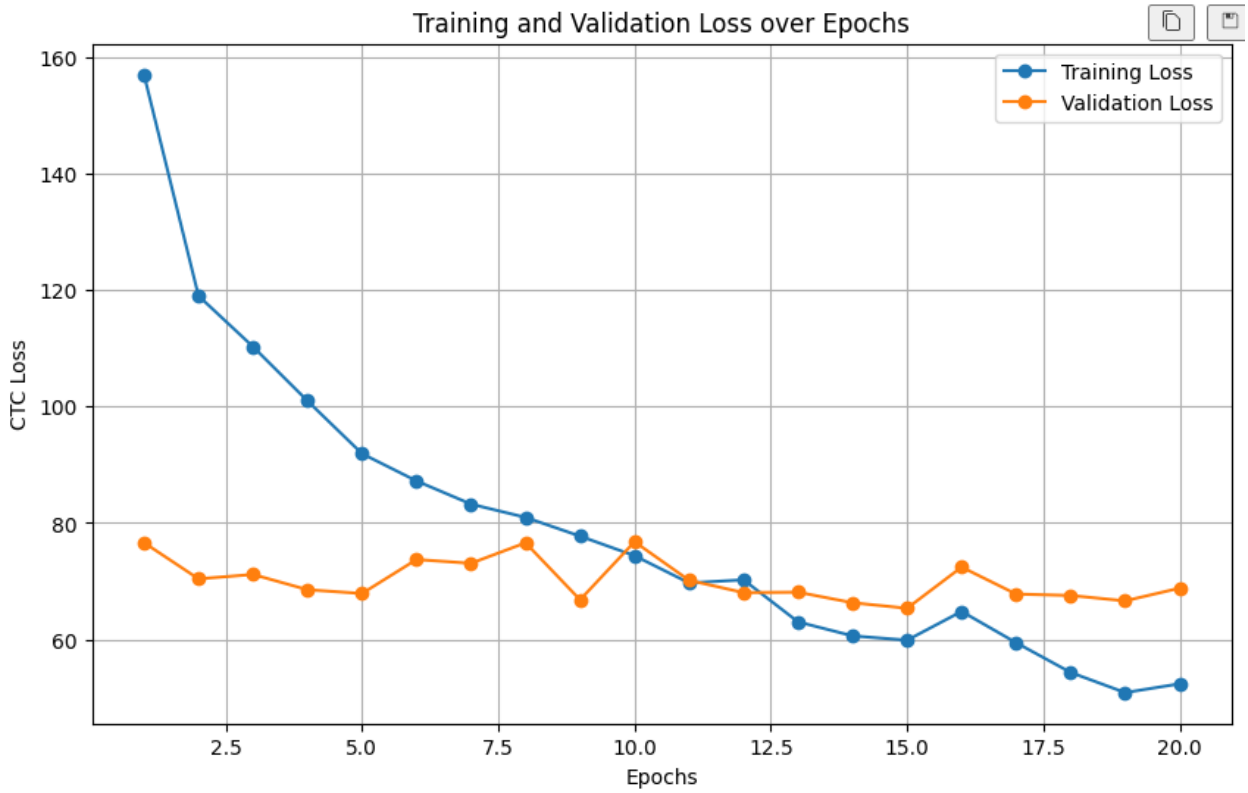
The table below describe why certain pre-packaged tools and hyperparameters were used for training.

Tool / Hyperparameter	**Description**
Wav2Vec2Processor	We use the default tokenizer and feature extractor provided by Wav2Vec2 as they are pre-tuned to handle typical speech-to-text tasks effectively. The processor ensures optimal pre-processing of audio and tokenization of text and is ready for use out of the box.
Wav2Vec2ForCTC	The Wav2Vec2ForCTC model is specifically designed for Connectionist Temporal Classification (CTC) tasks, which makes it ideal for end-to-end speech recognition. This model allows us to directly predict text sequences from audio inputs.
Learning Rate	1e-5 is used. This is a good starting point for finetuning models.
Batch Size	8 is used as a larger amount will trigger an out of memory error.
Epochs	30 is selected based on other online examples for finetuning audio models. Those models use smaller datasets of ~2k files, so we would expect our early stop to be triggered since we are using a significantly larger dataset.

Weight Decay	0.01 is applied to regularize the model by penalizing large weights, which helps prevent overfitting. This value is a common starting point in fine-tuning tasks and is effective for improving generalization.

2. Training Results Evaluation

The figure below plots the training and validation loss during the training process.



- Early stop was triggered after 20 epochs. We can see that after epoch 12, the model had started to overfit.
- The training loss steadily decreases after each epoch.
- The validation loss is relatively stable, with some fluctuation between the epochs. Furthermore, the validation loss does not decrease much. This indicates that the model performance on unseen data does not vary significantly.

2.1 Chosen Model

As we saved the checkpoints every 5 epochs, we will choose the model at epoch 15, since it only just started to overfit, and it still has a low validation loss.

3. Model Evaluation

The table below compares the pre-trained base model's performance against our fine-tuned model.

Metric	Base Model	Fine-Tuned Model	Improvement
Word Error Rate (WER)	0.108120	0.078751	0.029369
Character Error Rate (CER)	0.045444	0.034017	0.011427

3.1 Word Error Rate (WER)

We can see that the WER improved by 0.0294 (2.94%) after finetuning the model. This is a significant improvement in the performance of the model, since the base model was already performing quite well. Relative to the base model, the fine-tuned model's performance increased by 27.2% $(^{0.108120\ -\ 0.078751}/_{0.108120})$.

3.2 Character Error Rate (CER)

The CER also improved by 0.011427 (1.14%) after finetuning the model. This is a significant improvement, given that the base model has a very low CER of 0.045444. Relative to the base model, the fine-tuned model's performance increased by 25.1% $(^{0.045444\ -\ 0.034017}/_{0.045444})$.

4. Conclusion

In this training exercise, we have successfully fine-tuned the Wav2VEc2 model on a subset of the cv-valid-train dataset, resulting in significant improvements in both the Word Error Rate (WER) and Character Error Rate (CER). The fine-tuned model demonstrated a 27.2% relative improvement in WER and a 25.1% improvement in CER compared to

the pre-trained base model, highlighting the effectiveness of our fine-tuning approach. Our chosen model from epoch 15 provided an optimal balance between low validation loss and minimal overfitting, as evidenced by the training and validation loss trends.

5. Future Work To Improve Accuracy

We are limited in time and compute, if there is future opportunity to train the model again, we could try the following:

- Increase the regularization used (e.g. higher weight decay or introduce dropout)
- Implement learning rate scheduling to optimize the loss convergence
- Try different learning rate values
- Use a larger dataset for training (we are only using 5% of the data available)
- Data Augmentation such as injecting noise to improve the robustness of the model
- Look into layer freezing (further research will be required on the implications of freezing layers). From past experience in computer vision models, freezing appropriate layers improved performance.