1. Summary

Fine-tuned model performed **worse** than the original on the `cv-valid-dev` dataset with a Word Error Rate (WER) of **0.140** vs the baseline 0.110. After triage, among other possible improvement areas, the root cause is most likely due to the objective function only penalizing for the Connectionist Temporal Classification (CTC) loss, which **focuses on the "how it sounds like"** but **not considering the "how it is written"**, i.e. the language modelling side of things.

This report outlines the key observations made and discusses the implications, and then suggests some possible next steps to improve the accuracy.

2. Observations and Discussion

2.1. Relative performance on cv-valid-dev

Breaking down into categories, out of the 4075 data points,

	Number of rows
Both fully-correct (0 WER)	1713 (42%)
Both same non-zero WER	941 (23%)
Fine-tuned model is better	376 (9%)
Fine-tuned model is worse	*1045 (26%)*

Focusing on the *fine-tuned model is worse* case, I examined the **top 20 largest WER differences** between the 2 models (Refer to Appendix 1 - top 20 largest WER differences). **Across all cases**, the fine-tuned model was (1) **overly sensitive to audio artefacts / noise** in the mp3 and (2) **predicted characters that sounded like how it was pronounced instead of the English text** itself. For example, `cv-valid-dev/sample-003039.mp3` is by a British male, and the transcription was

"how did it start" → "HOW DIS STOUT"

(3) It also has a **lack of appreciation of the various accents**. For example, a German male:

> “This isn't a kidnapping” → “ZHIS IS UN'T A CAT NABAYN”

(4) **Missing/Inaccurate speaker metadata**. Another interesting find is that there exists cases where a data point claims that the speaker has a standard US accent, but the audio actually shows a different accent (e.g. `cv-valid-dev/sample-000630.mp3`). And not to mention the fact that majority of the speaker metadata are not provided (age, gender, accent). This suggests that the **training may have been overfitting towards the majority classes** (e.g. male with a US accent in his twenties), but we are unable to split the dataset more appropriately or apply differentiated handling (like undersampling the majority classes).

A brief analysis of the ***fine-tuned model is better*** case (Refer to Appendix 2 - top 10 cases where fine-tuned performed better) yielded particularly noteworthy other than a seemingly better performance of apostrophe prediction.

2.2. Analyzing the training process

The relatively worse performance was also evident on the `cv-valid-train` validation dataset, where the fine-tuned model also has a higher WER of 0.133 (vs 0.120). This suggests that the **problem manifested during training**.

Investigating deeper into the `Wav2Vec2ForCTC` model, I found that I was applying only the CTC loss for the loss function but had no loss component to account for the Language Modelling (LM) side of things.

To complicate matters, I froze all modules except the `lm_head`, the last module of the model. This module handles both (1) translations of the compressed audio features to the probabilities of a token sequence, including the language modelling aspect, as well as to some extent (2) task-specific domain adaptations. Since I did not penalize poor language modeling by using the vanilla `Wav2Vec2ForCTC`‘s CTC loss, as I trained, the module gradually lost sight of predicting with good language understanding.

The training metric graphs are consistent with the above. While both training loss and validation loss reduces with time steps, the WER seems to be on a downward (improve) trend initially until around step 800k, thereafter it worsens (Refer to `asr-train/notebooks/cv-train-2a.ipynb`). This may be a sign that the `lm_head` has lost its language model capabilities significantly.

3. Possible next steps

The main findings identified earlier are:

1. Model lacked respect for language modelling
2. Model was over-sensitive to audio artefacts / noise
3. Model could not handle minority class speaker profiles well (accent, gender, age)

This section proposes a series of steps to tackle these areas and improve accuracy.

3.1. Pre-training synthetic data curation

Focusing on the minority data classes, we can curate a **synthetic dataset that is well-balanced** and **covers a variety of speaker profiles / scenarios**. Here's how it goes:

Use text generation models/software (e.g. ChatGPT) to generate similar or a diverse set of utterances that our use case targets.

Use Text-to-Speech models/software (e.g. https://huggingface.co/voices or Amazon Lex) to read-out in a plethora of speaker profiles (accents, gender, age) and utterance profiles (pitch, tone, etc.), the target text (e.g. Common Voice dataset) to generate corresponding audio clips.

Use Speech Synthesis Markup Language (SSML) templates to provide some variations in the tone (e.g. asking tone, statement, longer pause than normal, 'er' and 'em's, etc.).

3.2. Data curation from the wild

Reach out to collaborate with Massive Open Online Courses (MOOC) to use videos and transcripts of lecturers with a variety of speaker profiles. Split videos based on the `.vtt` files, paying particular attention to accurate segmentation as subtitles tend not to align exactly with the speaker's utterances.

This **allows us to have access to real samples (with labels) of speakers in the wild, of various profiles**, in particular, **boosting the samples for minority classes**.

3.3. Predict speaker profiles for better minority class handling

Another approach to consider is to **predict a speaker's profile**. Knowing more speaker metadata will not allow us to perform more accurate stratified splitting, but also **facilitate techniques like oversampling / under-sampling** for neural network training, to ensure a balanced training.

We can start by predicting the accents of each of the Common Voice data. E.g. via https://huggingface.co/Jzuluaga/accent-id-commonaccent_xlsr-en-english . (NOTE: based on a few samples, this model doesn't seem very accurate).

3.4. Tune in-training data augmentation settings and add data augmentation

There are several techniques that can / should be applied on-the-fly as we yield training data for the model.

1. SpecAugment
    a. Wav2VecForCTC already applies this. But we need to tune the `mask_feature_prob` and `mask_time_prob` parameters.
2. Speed perturbation
    a. Increasing/decreasing the audio speech while normalizing pitch
3. Background noise (conversation, hum, pops and clicks, etc.)

We can leverage the `Optuna` experiment trials setup (as used in `cv-train-2a.ipynb`) to test on a smaller dev set to establish the optimal probabilities and speed perturbation range.

Data augmentation can help train our model to be **robust to perturbations in the incoming audio**.

3.5. Add LM score regularization term to loss function

This will ensure that during training, the model is penalized appropriately when it doesn't output probable English words (instead of phonetic gibberish). Refer to the code sketch below for a greedy **LM-score** validation loss computation.

An even simpler solution is to use **edit-distance** between the predicted text and the ground truth labels and add it as a regularization term. But there are corner cases where this may not be as appropriate. E.g. Speaker spoke 'stranger' but 'stronger' was predicted, which has vastly different meaning.

Adding a language model loss component should **regulate the ASR training** and guide it to predict proper English words, even if the decoding is greedy.

Moreover, this **should also make the model predict/react less to noise artefacts** as the noise tends to make the words sound gibberish.

```
def greedy_n_lm_score_val_loss(
   val_loader: DataLoader,
   w_lm_score: float,
) -> float:

   import kenlm  # type: ignore
```

```
    import torch
    from transformers import Wav2Vec2ForCTC, Wav2Vec2Processor

    model = Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-base-960h")
    processor = Wav2Vec2Processor.from_pretrained("facebook/wav2vec2-base-960h")
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    model.eval()

    lm_path = "common_crawl_5gram.arpa"  # TODO replace
    lm_model = kenlm.Model(lm_path)

    preds = []
    labels = []
    val_loss = 0.0
    for batch in val_loader:
        with torch.no_grad():
            out = model(batch["input_values"].to(device),
labels=batch["labels"].to(device))

        # Greedy decoding
        predicted_ids = torch.argmax(out.logits, dim=-1)
        transcription = processor.batch_decode(predicted_ids)
        # Assume higher is better
        # lm_score = lm_model.score(transcription)
        # acoustic loss + w_lm_score * LM loss
        val_loss += out.loss.item() + w_lm_score * lm_score
        preds.extend(transcription)
        labels.extend(processor.batch_decode(batch["labels"],
group_tokens=False))

    val_loss /= len(val_loader)

    return val_loss
```

3.6. Add beam search with LM scoring during inference

Use a beam search decoder like `ctcdecode.CTCBeamDecoder`, and apply **LM scoring** to get the **most plausible English sequence** amongst the beam candidates. This trades a longer runtime and more compute resources for more accuracy.

3.7. Fine-tune with curriculum learning

The current training regime only fine-tunes the `lm_head`. While this is ok, the low number of parameters (<1M) may not have sufficient representation capacity to handle both language modelling responsibilities on top of domain-specific quirks (e.g. accents, gender, age). We want `lm_head` to focus on acoustic to language translations.

To **avoid `lm_head` from drifting to learn acoustic features**, we can keep it frozen and train the `encoder` module for 1-2 epochs. Then unfreeze it to train both modules fully.

To be able to train the `encoder` part as well on my low-powered 11GB GPU, I will need to experiment with even batch sizes that are even lower than 8, and use **gradient accumulation** techniques to be able to do achieve effective gradient updates as a typically effective batch would be (e.g. to achieve a 32 effective batch size using a size of 4, accumulate gradients for 8 times before running an optimizer step).

4. Appendix

Appendix 1 - top 20 largest WER differences

filename	wer_diff	is_us_accent	accent	gender	text	prediction	comment
cv-valid-dev/sample-000104.mp3	1	non-US	UK	male	i'll hang it up	O HI I DOT	Noisy background, but clear voice. Model predicted phonetic gibberish.
cv-valid-dev/sample-002409.mp3	0.78	non-US	IN	male	wouldn't it be nice if we were like that	UDEN'TER GENICE A WIG OR LIKE THAT	Mumbling. Model predicted phonetic gibberish.
cv-valid-dev/sample-002676.mp3	0.78	non-US	AU	male	half an hour later his shovel hit something solid	HALF A NAIL LIHTAHE' SHEVEL HEAT'S SOMETHING SULLID	Model predicted phonetic gibberish.
cv-valid-dev/sample-001309.mp3	0.75	non-US	IN	male	that was his work	DECK WASH HISWURK	Model predicted phonetic gibberish.
cv-valid-dev/sample-003717.mp3	0.75	US	US	male	they said nothing else	THAY SAT MASSHING ELSE	Model predicted phonetic gibberish.
cv-valid-dev/sample-003419.mp3	0.75	US	US	male	who's the ten for	RUSED THE TENFOR	predicted phonetic gibberish. Noise at the beginning was associated with a 'R' sound.
cv-valid-dev/sample-002561.mp3	0.75	non-US	DE	male	this isn't a kidnapping	ZHIS IS UN'T A CAT NABAYN	Model predicted phonetic gibberish.
cv-valid-dev/sample-003039.mp3	0.75	non-US	UK	male	how did it start	HOW DIS STOUT	Model predicted phonetic gibberish.

cv-valid-dev/sample-003864.mp3	0.67	non-US	UK	male	i've got a flash for you	O GOOT AR FLESH WE YEE	Muffled sound. Model predicted phonetic gibberish.
cv-valid-dev/sample-000071.mp3	0.67	non-US	FR	male	who do you think writes them	E DIE THING WRITES THEM	Model predicted phonetic gibberish.
cv-valid-dev/sample-000958.mp3	0.62	non-US	ES	male	so you know about love the boy said	O YOU NOBLE LOVETHE BOY SAID	Model predicted phonetic gibberish.
cv-valid-dev/sample-002759.mp3	0.6	non-US	unk	male	take hold of the cover	A COM OF TECAVER	Model predicted phonetic gibberish.
cv-valid-dev/sample-003572.mp3	0.6	non-US	IT	male	that first day everyone slept from exhaustion including the englishman	DAT FIRST DEY EVER WONE A SLIPPED FROMEXOCLUTINGIN	Data was marked as US accent, but not really. Model predicted phonetic gibberish.
cv-valid-dev/sample-000214.mp3	0.6	non-US	IN	female	and we had a bite	I BE HI O BITE	Model predicted phonetic gibberish.
cv-valid-dev/sample-003944.mp3	0.6	US	US	male	get a load of this	GET EM OVOR THIS	Audio artefacts causing voice to be break up. Model predicted phonetic gibberish.
cv-valid-dev/sample-003722.mp3	0.6	non-US	unk	male	the city was still sleeping	SITHEUS STILL SLEEPING	Model overly sensitive to breathing. Model predicted phonetic gibberish.
cv-valid-dev/sample-003558.mp3	0.6	non-US	unk	male	get martin out of jail	ET MARTIN AUT OF TRAIL	Model predicted phonetic gibberish.
cv-valid-dev/sample-000057.mp3	0.59	non-US	CA	female	before guns were invented armies had to throw bullets at each other and if a bullet touched you you had to sit out	OM THAT TO THROW A WHILL IT TOUCHED YOU YOU HAD TO	Model overly sensitive to noise. Model predicted phonetic gibberish.

					until the next war		
cv-valid-dev/sample-000630.mp3	0.55	non-US	unk	male	think i'll go home and see what the family is doing	ING A'LLGO HOMEAND SEE WHAT THE FEMILY IS DOING	Data was marked as US accent, but not really. Model predicted phonetic gibberish.
cv-valid-dev/sample-003189.mp3	0.5	non-US	DE	male	it seemed such a little thing so bright and small and still	ID SEEM IT TOUCH A LITTLE THINGSO BRIGHT AND SMALL AND STILT	Model overly sensitive to noise. Model predicted phonetic gibberish.

Appendix 2 - top 10 cases where fine-tuned performed better

1. `cv-valid-dev/sample-001346.mp3` -> (US; US; male)
2. `cv-valid-dev/sample-003047.mp3` -> (US; US; male). 20+s audio had additional conversation which was picked up fairly well by the original model, but is not the provided ground truth.
3. `cv-valid-dev/sample-002782.mp3` -> (US; US; male). Model was able to predict correct locations of `'` and the word "invasion". 20+s audio with additional conversation.
4. `cv-valid-dev/sample-001352.mp3` -> (non-US; UK; male). Model is able to predict "I'm" instead of "I AM" by the original model.
5. `cv-valid-dev/sample-001015.mp3` -> (non-US; UK; male). Model was able to get "NOT" and "SO" correct over the original model.
6. `cv-valid-dev/sample-000606.mp3` -> (non-US; unk; male). Model was able to predict "I'm" and correct words.
7. `cv-valid-dev/sample-000723.mp3` -> (non-US; unk; male). Model was able to predict "I'll".
8. `cv-valid-dev/sample-002313.mp3` -> (non-US; unk; male). Model was able to predict "LET'S" (original predicted "LAT'S").
9. `cv-valid-dev/sample-004001.mp3` -> (non-US; UK; male). Proper English audio. Model was able to predict correctly.
10. `cv-valid-dev/sample-003659.mp3` -> (non-US; unk; male). Muffled audio, heavily accented. Model was able to predict correctly.

Side note - NaN/infinite validation loss for 1 batch

Side note, for full transparency, The following batch of mp3 files resulted in NaN/infinite validation loss, which may yield other clues as to poor training.

- cv-valid-train/sample-075795.mp3
- cv-valid-train/sample-023363.mp3
- cv-valid-train/sample-093628.mp3
- cv-valid-train/sample-014576.mp3
- cv-valid-train/sample-074754.mp3
- cv-valid-train/sample-145014.mp3
- cv-valid-train/sample-122947.mp3
- cv-valid-train/sample-062523.mp3

Listening to their audio, there was nothing in particular that stood out. It could be intermediate processing that’s causing an issue. An open question to find out.