This section covers all the errors encountered, confusing steps, time tracking, GPU requirements, and final audio quality after fine-tuning a voice cloning model.

1. Errors Encountered

a. Hardware Compatibility

- **Error**:
  NotImplementedError: Unsloth currently only works on NVIDIA GPUs and Intel GPUs.
- **Cause**: Attempted to run on unsupported hardware (e.g., AMD GPU or non-GPU runtime).
- **Fix**: Switched to a supported runtime with an NVIDIA Tesla T4 GPU on Google Colab.

---

b. CUDA Memory Issues

- **Error**:
  RuntimeError: CUDA out of memory. Tried to allocate 70.00 MiB...
- **Cause**: Batch size and model weights exceeded available VRAM (~14 GB).
- **Fix**:
  - Lowered batch size.
  - Used gradient accumulation.
  - Cleared cache between runs with torch.cuda.empty_cache().

---

c. Data Type Mismatch

- **Error**:
  RuntimeError: Input type (torch.cuda.FloatTensor) and weight type (torch.cuda.HalfTensor) should be the same.
- **Cause**: Mismatch between model precision and input tensor.
- **Fix**: Made input tensors match model weights by ensuring uniform data types using .half() or .float() across the pipeline.

---

d. PyTorch Version Compatibility

- **Error**:
- ERROR: Could not find a version that satisfies the requirement torch>=2.7
- **Cause**: Torch 2.7 not available in pip index at the time.
- **Fix**: Installed the latest available version (torch==2.6.0) that was compatible with current system and dependencies.

I was unable to log all the errors because I did not read the instructions carefully. I sincerely apologize for the oversight.

2. Confusing Steps

Step	**Description**
Installing Dependencies	Conflicts between required versions of torch, transformers, and unsloth.
Batch Size & Hyperparameters	Finding the optimal batch size that fits in GPU memory without causing OOM errors.
Using NLTK	Initially assumed NLTK was needed for sentence splitting, but Whisper already provided sentence-level segments in its output JSON.
Voice Cloning Inference	Limited GPU quota on Google Colab Free Tier resulted in slower inference and reduced output quality.
AWS vs Colab	AWS instances didn't support the setup as smoothly due to driver/library mismatches and required more manual configuration compared to Colab.

3. Time Taken for Each Section

Data Collection and Prepration – 3 hours

Finetuning and inference – 4 hours

4. GPU/Resource Requirements

Resource	Details
GPU Used	NVIDIA Tesla T4
VRAM	14.741 GB
Platform	Google Colab (Free Tier)
Frameworks	PyTorch, Whisper, Unsloth
Sampling Rate	24000 Hz

5. Final Audio Quality Achieved

Audio Clarity	Medium – Audio generated was intelligible but lacked natural prosody.
Issues Faced	Due to limited GPU and shorter inference time, audio had robotic undertone.
Sampling Rate	24,000 Hz
Reason for Degradation	Inference run on low-resource environment (Colab Free Tier with quota limits).

FINAL REPORT:-

A custom dataset was created and successfully pushed to Hugging Face under the repository: gary7372/voice_clone_sample_24000hz. The audio used for this dataset was extracted from a 58-minute long video, which was the same video provided on YouTube for the assignment.

To obtain transcriptions from the audio, OpenAI's Whisper model was utilized. Instead of using NLTK to break down the transcriptions into sentences, Whisper's default output format, which already includes segmented sentences in its JSON output, was used. This eliminated the need for additional sentence segmentation using external tools like NLTK.

Once the transcription data was ready, the audio was chunked into smaller segments aligned with the corresponding transcriptions. These audio-text pairs were then used to construct the dataset. After proper formatting and validation, the dataset was uploaded to Hugging Face for public access.

During the model training phase, I initially faced challenges with GPU memory limitations. The selected hyperparameters were too demanding for the available hardware, causing the model to fail to fit into GPU memory. After several iterations and adjustments to batch size and model configuration, I was able to initiate the training process successfully.

However, during inference, the limitations of the free-tier Google Colab environment became evident. Due to restricted GPU availability and processing power, the quality of the generated audio was not up to expectations. The output audio was generated at a sampling rate of 24,000 Hz, in alignment with the dataset's configuration.

Suggestions for Improvement

- Use Higher-Tier GPUs: Training and inference on larger models require high-memory GPUs. Consider upgrading to a Colab Pro account or using alternative platforms such as Kaggle, AWS, or Azure for access to more robust hardware.
- Training Hyperparameters Optimization
- Dataset Expansion: Increasing the dataset size with more diverse and well-labeled samples could improve model generalization and output quality.
- Experiment with Alternative Models

Training on a small dataset significantly impacts a model's ability to generalize, often leading to overfitting where the model memorizes specific patterns rather than learning generalizable features. This can result in poor performance when handling longer or more complex sentences, unfamiliar words, or varied syntactic structures. Consequently, the generated speech may sound robotic, monotonous, or contain mispronunciations due to limited exposure to diverse phonemes and contexts. Additionally, with inadequate data, the model struggles to learn natural prosody and intonation, often producing speech with flat delivery, slurred syllables, or missing articulation. Expressive features such as pitch variation, emphasis, and emotional cues—essential for realistic and engaging speech—are typically absent or poorly replicated, causing the audio output to lack naturalness and emotional depth.