Audio CAPTCHA Transcription Report

Methodology and Preprocessing Steps

The primary goal of this project was to transcribe audio CAPTCHAs into text using Whisper. Below are the key steps undertaken:

1. **Dataset Handling:**
   - The dataset consists of 10,000 audio CAPTCHA files and corresponding images.
   - Due to computational constraints, a subset of 500 audio CAPTCHA files was selected for processing.
2. **Preprocessing Steps:**
   - **Loading Audio:** Each file was loaded at a sampling rate of 16 kHz using `librosa`.
   - **Noise Reduction:** Applied `noisereduce` to enhance clarity and improve transcription accuracy.
   - **Normalization:** Audio was normalized to ensure consistent amplitude across all samples.
   - **Feature Extraction:**
     - **MFCCs (Mel-Frequency Cepstral Coefficients):** Extracted to represent important speech features.
     - **Spectrograms:** Derived using Short-Time Fourier Transform (STFT) to analyze frequency patterns.
3. **Model Selection:**

   Multiple models were tested to determine the best approach:

   - wav2vec2
   - Conqui SST
   - OpenAI's Whisper
   - Speech2Text

   Each model was evaluated on a sample of 500 CAPTCHAs. **OpenAI's Whisper** outperformed other models in accuracy, though it struggled with correctly identifying numbers. To mitigate this issue, preprocessing techniques such as noise reduction and normalization were applied.

4. **Transcription:**
   - Whisper's "Medium" model was used for transcribing audio into text.
   - Outputs were formatted into "capital X, small y" notation for readability.

Challenges Encountered and Solutions

1. **Noise and Distortion in Audio Captchas**
   - Some CAPTCHA files contained significant background noise.
   - **Solution:** Applied `noisereduce` to enhance signal clarity before feeding it into Whisper.
2. **Processing Large Dataset**
   - Running Whisper on 10,000 files would have been computationally expensive.
   - **Solution:** Sampled the first 500 files for evaluation.
3. **Text Formatting for Better Readability**
   - Whisper's raw output was not structured for CAPTCHA-like transcriptions.
   - **Solution:** Applied post-processing to format output into "capital X, small y" format.

Evaluation and Error Analysis

1. **Manual Evaluation Approach:**
   - Due to the lack of ground truth labels in a digital format and the computational limitations of Google Colab's free tier, Optical Character Recognition (OCR) was not used to extract the ground truth labels.
   - Instead, manual verification was conducted by comparing the transcribed text with the corresponding CAPTCHA images.
2. **Observations:**
   - The model performed well on clear speech but struggled with distorted audio.
   - Capitalization errors were common, requiring post-processing adjustments.
   - Some characters (e.g., "O" vs. "0") were misclassified, leading to minor transcription errors.

Potential Improvements and Limitations

Potential Improvements

1. **Fine-Tuning Whisper on CAPTCHA-Specific Data**

   ○ Training Whisper on a labeled CAPTCHA dataset could improve recognition accuracy.
2. **Advanced Noise Filtering**

   ○ Experimenting with different noise reduction techniques (e.g., spectral subtraction) could further enhance transcription quality.
3. **Contextual Post-Processing**

   ○ Implementing rule-based corrections for commonly misrecognized characters (e.g., "O" vs. "0").

Limitations

- **Dependence on Whisper's Pretrained Model:** Since Whisper is not explicitly trained on CAPTCHA audio, it may not generalize perfectly.
- **Computational Constraints:** Processing the full 10,000 dataset would require significant computational resources.
- **Potential Misinterpretation of Special Symbols:** If CAPTCHAs include symbols or numbers, additional processing would be needed.

Conclusion

This project transcribed audio CAPTCHAs using Whisper while applying noise reduction and feature extraction. Future work could focus on fine-tuning models, expanding dataset processing, and improving character recognition accuracy.

AI Acknowledgment

During the development of this project, AI-powered assistants, including **ChatGPT, Claude, and DeepSeek**, were used to aid in writing, debugging, and optimizing the code. These tools provided support in:

- **Code Assistance:** Generating and refining Python scripts for preprocessing, feature extraction, and transcription.
- **Debugging:** Identifying and fixing errors in the implementation.
- **Documentation:** Structuring the report, README, and refining explanations for clarity.

While AI-assisted tools played a role in enhancing efficiency, all final decisions, modifications, and validations were performed manually to ensure accuracy and alignment with project requirements.