CAPTCHA Text Recognition Using Whisper AI And OCR

1. METHODOLOGY

➢ Preprocessing of Audio Data

At first, I tried to enhance the accuracy of the speech recognition model, that's why, I implemented a comprehensive preprocessing pipeline for the audio files. The primary goals of preprocessing were:

1. **Volume Enhancement**: Some CAPTCHA audio files had low volume, making transcription difficult. To counter this, I applied a **volume boost** of **1.5x** to ensure clarity in speech recognition.
2. **Normalization**: Audio normalization was used to standardize the amplitude levels across all files. This prevents volume discrepancies between different samples and ensures consistent model performance.

➢ Speech Recognition with Whisper AI

This project utilizes **Whisper AI**, an advanced automatic speech recognition (ASR) model developed by OpenAI. Specifically, I tested two different versions of Whisper to evaluate performance across different model sizes:

Model Size	Parameters	Size
Whisper Base	**74M**	**~140 MB**
Whisper Medium	**769M**	**~1.5 GB**

- The **Whisper Base** model is a lightweight version that offers faster inference speeds, making it suitable for real-time applications or environments with limited computational resources.
- The **Whisper Medium** model, being significantly larger, provides better transcription accuracy but requires more GPU memory.

Each audio file was transcribed using the following steps:

1. **Loading the Processed Audio**: I first loaded the audio files using the Whisper API.
2. **Running Inference**: The audio was passed to the Whisper model for transcription. The model internally converts the waveform into text using a deep learning-based sequence-to-sequence approach.
3. **Post-Processing**: Since Whisper sometimes misinterprets words, I applied post-processing techniques:
    - Converting number words to digits (e.g., **"two" → "2"**)
    - Fixing common misheard words (e.g., **"mall" → "small"**)
    - Handling capitalization errors (e.g., **"capital A" → "A"**)

There were many other post-processing modifications implemented on Whisper's primary output other than these, which ensured that the final transcriptions were as accurate as possible before being compared with the ground truth.

➢ Ground Truth Text Extraction Using OCR

To evaluate the accuracy of Whisper's transcriptions, I needed a reliable way to extract the **ground truth text** from CAPTCHA images. For this, I used **EasyOCR**, an optical character recognition (OCR) tool capable of detecting and reading text from images.

The OCR process was implemented as follows:

1. **Loading the Image**: Each CAPTCHA image file was loaded into EasyOCR's OCR pipeline.
2. **Text Detection**: The model first scanned the image and identified potential regions containing text.
3. **Text Extraction**: Once the text-containing regions were detected, EasyOCR extracted the characters from these regions.
4. **Concatenation of Results**: Since EasyOCR may detect multiple segments of text, I concatenated them into a single string to obtain the final extracted CAPTCHA text.

2. CHALLENGES FACED

I encountered several challenges, ranging from model selection issues to handling Whisper AI's raw outputs. Below, I describe these obstacles and the approaches taken to address them.

- **Selecting the Right Speech-to-Text Model**

Initially, I experimented with multiple speech-to-text models before settling on **Whisper AI**. The journey involved several failed attempts with other models:

DeepSpeech: This was one of the first models I considered due to its open-source nature and relatively good performance on various speech datasets. However, I quickly discovered that DeepSpeech was not accessible on Kaggle due to installation and compatibility issues. The necessary dependencies and TensorFlow versions required to run DeepSpeech could not be properly installed within Kaggle's environment.

Wav2Vec2: I then attempted to use Wav2Vec2, which is a powerful model developed by Facebook (Meta) for speech recognition. While Wav2Vec2 is known for its accuracy, it is also significantly larger in size and requires extensive computational resources. Running inference on pre-trained models was manageable, but training or fine-tuning on Kaggle took an excessive amount of time.

I also attempted to use several python built-in libraries like speech-recognition, Vosk etc. but also faced issues there. After these setbacks, I decided to switch to **Whisper AI**, developed by OpenAI. It provided a balance between accuracy, speed, and ease of implementation, making it the best choice given my computational constraints.

- **Handling Whisper AI's Raw Output**

Although Whisper AI performed well in terms of transcription accuracy, its raw output format posed significant challenges. The transcriptions it produced were often inconsistent with the expected CAPTCHA format. Instead of direct alphanumeric strings, it generated outputs like: SMALL GEE CAPITAL VE FOUR CAPITAL A CAPITAL C NINE, which should have been: gv4AC9. To fix these, I had to implement several post-processing modifications to make it generate the text in a way I want.

- **GPU Memory Issues Due to Audio Preprocessing**

The most significant challenge I faced in this project was managing GPU memory usage when processing the audio files. To improve Whisper AI's transcription accuracy, I initially attempted to enhance the quality of the input audio through normalization, noise reduction, and volume amplification. However, these steps led to extreme GPU memory usage, particularly when I applied a high-volume amplification.

At first, the combination of high-volume amplification and normalization significantly increased the size of the audio tensors, causing out-of-memory (OOM) errors during Whisper's processing. After several attempts to optimize memory usage and fine-tuning, I was able to resolve the issue by adjusting the volume amplification to 1.2, which helped reduce memory consumption without compromising the quality of the audio too much. With this change, I was able to process the files without encountering further GPU memory issues.

3. RESULTS

Model Configuration	Total Characters Accuracy Match	Total Matched Texts	Probability of Total Matched Texts
Whisper Base (Before Pre-Processing Audio Samples)	**73.1% (**± 1)	**2542/10000**	**0.254**
Whisper Medium (Before Pre-Processing Audio Samples)	**79.1% (± 1)**	**2962/10000**	**0.296**
Whisper Medium (After Pre-Processing Audio Samples)	**79.5% (± 1)**	**761/2500**	**0.304**

This section presents the accuracy and performance of Whisper AI's speech-to-text models when tested on the CAPTCHA dataset under different conditions.

The Whisper Base model, with no pre-processing applied, achieved an accuracy match of 73.1%. In comparison, the Whisper Medium model, also without pre-processing, demonstrated a noticeable improvement, reaching an accuracy of 79.1% with 2,962 correct matches out of 10,000 samples.

However, the most notable improvement came from Whisper Medium after applying pre-processing techniques to the audio samples. With a slight enhancement in the audio quality through normalization and noise reduction, Whisper Medium reached an accuracy match of 79.5%, marking a further improvement in transcription quality. Despite the smaller increase, it showed the most promise, with 761 out of 2,500 samples accurately transcribed, indicating the model's sensitivity to better-quality audio inputs.

Audio File	Whisper AI Transcription	Ground Truth (OCR)	Match
captcha_9452.wav	smQDbA	smQDbA	✔
captcha_9206.wav	sP5kpa	sPSkpa	X
captcha_1755.wav	6ajmJy	6ajmJy	✔
captcha_9501.wav	KZiOKC	KZiOKC	✔
captcha_3590.wav	vxClFA	dxCIFA	X
captcha_8609.wav	KyUvqA	KyUvqA	✔
captcha_8114.wav	y0czO	YOczzo	X
captcha_7849.wav	DFFU9S	DFFU9S	✔
captcha_0049.wav	4OPnyT	4OPnyT	✔
captcha_2228.wav	ITocLH	ITocLH	✔

This Table shows the result of 10 random samples generated from Whisper Medium model on pre-processed samples. The results indicate that although only 7 out of 10 samples were perfectly matched, a closer look reveals that almost all the unmatched predictions have just a single-character difference from the ground truth. This strongly supports my earlier statement that, on average, 5 out of 6 characters are being correctly predicted by Whisper AI.

Additionally, it's important to note that some discrepancies may also arise from OCR errors, as the model occasionally struggles to differentiate between visually similar characters such as S and 5, I and l, and 8 and B. This suggests that the actual performance of Whisper AI may be even better than what is reflected in the raw accuracy scores.

4. IMPROVEMENTS

Applying pre-processing techniques, such as normalizing volume levels and reducing noise, could have significantly improved transcription accuracy with Whisper. Although GPU limitations initially hindered the testing of pre-processed samples, I was able to resolve the issue by adjusting the volume to a lower level of 1.2. This change allowed the model to process the data effectively, and I believe that with further optimization of the audio, Whisper AI could have delivered even higher accuracy with pre-processed samples.

Additionally, while Whisper Medium showed promising results, it is not the most advanced version available. Whisper Large, with its larger number of parameters, could have provided even better results. However, due to computational constraints, Whisper Medium was chosen. With more GPU resources, utilizing Whisper Large could have significantly boosted transcription accuracy, though it would have required more time and power to train.

Another area for improvement would be training the audio as spectrogram images and pairing them with ground truth texts for use with larger pretrained models like Wav2Vec 2.0 or DeepSpeech. These models have demonstrated strong performance in speech-to-text tasks and, with proper fine-tuning, could yield higher accuracy. While these models require significant resources and extended training times, they offer promising avenues for further improving transcription quality.