Performing Under (com)Pressure: Evaluating the Accuracy of Whisper on Speech Passed Through Different Audio Codecs

Tianqi Zhang

1 - Introduction

1.1 - Speech to Text

Speech-to-text has changed how many people live their lives today. Whether they are asking Siri to set a timer, interacting with a virtual assistant over the phone, or texting a friend while washing the dishes, most people with regular access to technology will have interacted with some sort of speech-to-text model. With the advent of neural networks, these models have become inconceivably powerful and incredibly robust. However, there is no free lunch: these models are also exceptionally expensive to build and to run, too expensive for most people and even smaller companies/organizations. Today, many of the best, most cutting-edge models are also closed source, making it impossible to run locally. Thus, most users will be interacting by sending data to a server somewhere else to run the model there. As a result, the transferring of data becomes a massive bottleneck for people who want to use these models but do not have the resources at home to run them. Minimizing the size of audio files to limit the overhead for transferring data and get faster transcription becomes an incredibly important task.

1.2 - Compression

Compression is the practice of reducing the size of a file for faster transfer or more efficient storage. While lossless compression methods retain all the original data, there are limits to how much they can reduce the size as a result. Meanwhile, lossy methods find smart ways to retain and encode the important data while throwing away the data that is less important. How each method decides what's important and how the important data is encoded varies, and can have a large impact on how "good" the output sounds, especially in the presence of environmental noise. Thus, it is easy to see how finding a method that can minimize file size while still retaining the information that is important to a speech-to-text model has important applications for designers and users of speech-to-text software.

1.3 - This Project

This project explores the impact of different methods of input data compression on the performance of an automatic speech recognition model by using different codecs to compress and decompress speech data, then measuring the performance of a flagship ASR model on them. In this paper, I will first describe the setup of the project, including the tools used, data used, and methodology. I will then present the results along with interpretations. Then, I will discuss some limitations of this project that are important to understand for the conclusion. Finally, I will give my conclusion and recommendations for future research.

2 - Tools and Data

For this project, I used OpenAI's Whisper as the ASR model to evaluate. The compression codecs I used were mp3, Opus, and Opus for voice, as well as downsampling for a baseline/worst case. I also used FFmpeg to compress and decompress speech and Librosa, SoundFile, and NumPy for handling audio in Python. I used JiWER to calculate the word error rate to evaluate Whisper's performance. Finally, I used Pandas and Seaborn to compile the results and make visualizations of the data I collected.

For the data, I used audio files of speech from the Librispeech ASR corpus (specifically their clean test set) and their corresponding ground-truth text files (Panayotov et. al.). I specifically used speakers 61 and 121, a male and a female voice respectively. I used one speech file from each speaker for a total of 2 files.

I also used noise from the DEMAND (Diverse Environments Multichannel Acoustic Noise Database), which can be found in this Kaggle dataset. For this project, I specifically chose noise that would be common for someone using speech-to-text while out and about, picking park noise for its ambiance and noise from a meeting to test performance in cases of overlapping speech.

The specific files used can be found in this Google Drive folder.

3 - Hypotheses

The initial hypothesis was that a codec optimized for speech, namely Opus for voice, would be able to compress speech the best without compromising the performance of the ASR model. I reasoned that a codec built specifically for speech audio would be able to capture important speech characteristics better than a general audio codec, so it would emphasize the things an ASR model would be looking out for. I also hypothesized that this would especially be the case for speech audio with environmental noises, as while general compression methods would get caught up trying to retain elements of the noise, Opus voice would know to keep only the speech elements at lower bitrates.

4 - Methods

To test these hypotheses, I first added the different types of noise to each speech file using Python. I chose two separate signal-to-noise ratios (SNRs), 10dB and 20dB, to account for different environmental conditions. The signals and noises were then loaded into NumPy arrays using Librosa, which were then added, with a multiplier applied to the noise to achieve the desired SNR. This results in 10 audio files: one for each noise type (park and meeting) for each SNR (10dB and 20dB) for each of 2 speech files, plus 2 "clean" files with no noise added.

Then, I used FFmpeg to compress each audio file using each codec at different bitrates. The bitrates I chose for this project were 128, 64, 32, 16, and 8 kbps. The codecs and compression methods I chose, as mentioned above, were mp3, Opus, Opus for voice. I then used FFmpeg again to decompress these files back into WAV files. I also used downsampling for

compression. Downsampling is the act of simply resampling an audio file at a lower sampling rate. This reduces size, but is indiscriminate about what information is kept/thrown out, making the quality of the output files much worse than a codec that is smarter about what to keep. Thus, downsampling a file to an equivalent bitrate served as an effective baseline for this project, as any good codec should easily outperform downsampling. To calculate the new sampling rate, I used the following equation, assuming a 16-bit bit depth for all the original files:

$$\text{bitrate} = \frac{\text{sampling rate} \times \text{bit depth}}{1000}$$

I was left with a final dataset of 200 files: One for each of 5 bitrates, 4 codecs/methods, and 10 uncompressed files.

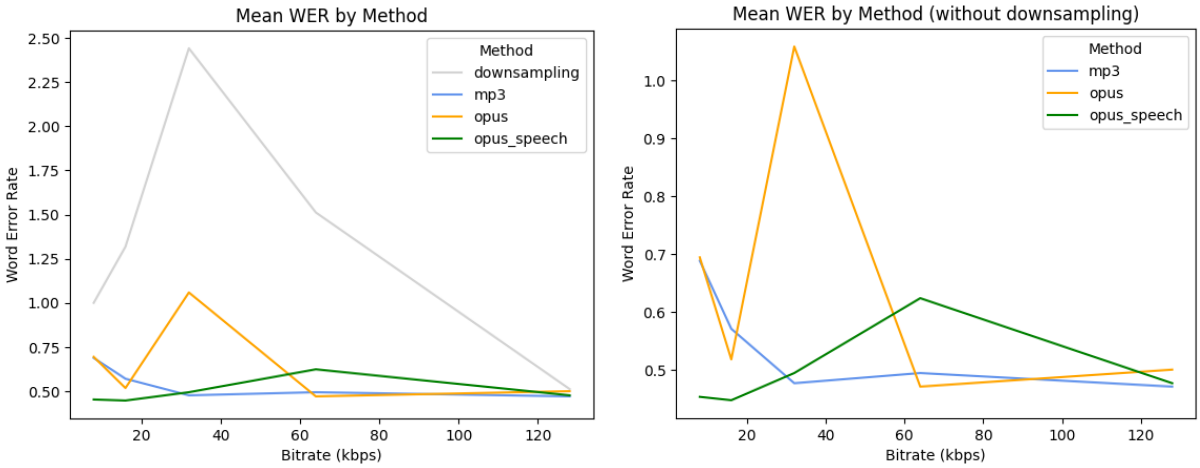
Finally, I passed each of these 200 files through Whisper and measured its performance on them. For this project, I used the "base" size model, which boasts 74 million parameters. I specifically used the multilingual model, so the results of this project could generalize a bit better and the code could be easily reused for testing another language. For evaluation, I used word error rate (WER) as the performance metric. To do this, I installed Whisper locally, then wrote scripts to automatically pass each file through the model and obtain the output. I then used code from this Medium article (Marangon, 2023, paras. 16-17), using JiWER to calculate the WER from a reference and a hypothesis (the output of the model). I compiled my results in a Pandas dataframe, with one row for each file and columns containing the file name, speaker, noise added, codec used, SNR, bitrate, and finally the WER. All my code can be found in this public GitHub repository.

5 - Results

Some quick notes on this section: I will be referring to the "performance of the codec" a lot in this section. Keep in mind what is actually being measured is Whisper's performance, evaluated using WER. This is then used to determine which compression method better retains the important features of the speech. "Opus voice had a better performance" and "mp3 achieved this WER", for example, are shorthands for "Whisper performed with a lower WER on the files compressed using the Opus voice codec" and "Whisper achieved a WER of this on a file compressed with mp3".

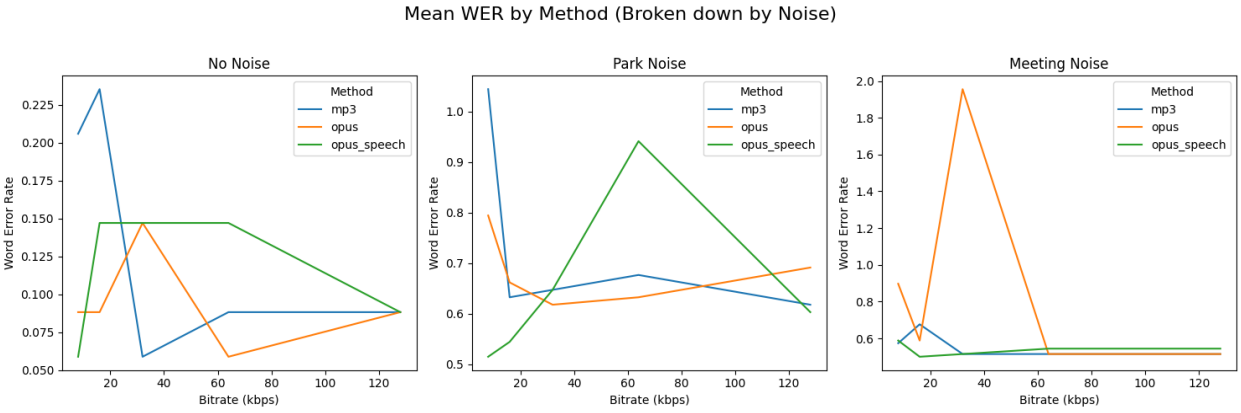
There will also be WERs that are above 1 shown in the graphs. This is possible because the total number of substitutions, deletions, and insertions in the hypothesis could be more than the number of total words in the reference.

5.1 - Overall Results



These two graphs show the averaged WER by bitrate for each compression method. For these, the right graph has downsampling removed so that the difference between each codec can be more easily seen. At higher bitrates, it seems like mp3 slightly outperforms the other codecs. However, at the lowest bitrates, anything below about 30 kbps, Opus for voice wins by a significant margin, beating out the second best, mp3, by 0.23 at 8kbps. Since we want higher performance at lower bitrates, Opus seems to give the best performance, even doing better at 8 kbps than opus and mp3 at 64 kbps. This is likely because, as hypothesized before, Opus for voice assumes speech-like input, so it opts to keep more features that are important for speech. This means that, after compression, there are more speech-specific features left, making it easier for Whisper to transcribe.

5.2 - Results by Noise Type



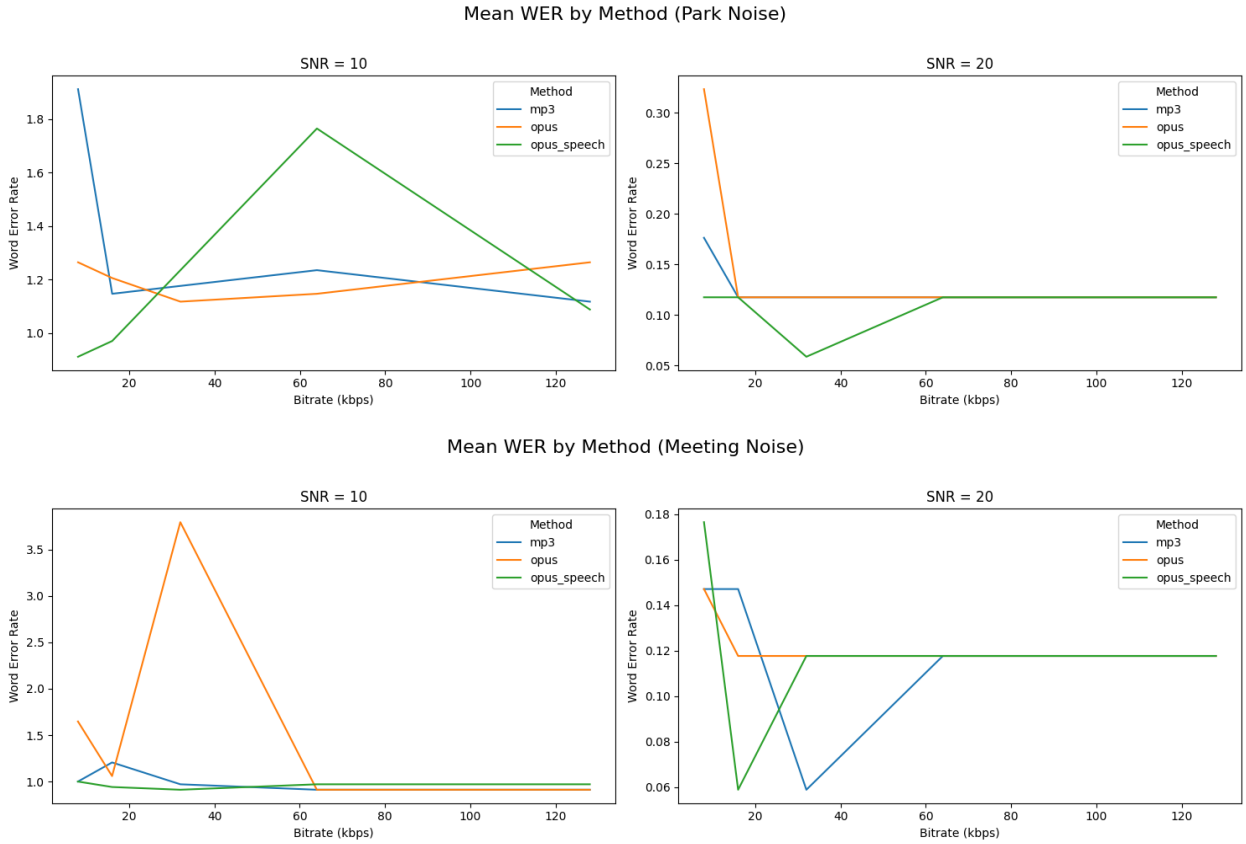
For these graphs, the WER is averaged between the two SNRs of 10 and 20. Here, it is evident that, at the lower bitrates, Opus for voice drastically outperforms mp3 and general-audio-Opus in conditions of no noise and of environmental noise (park noise). With environmental noise, this is likely because Opus voice will retain more speech features and less features from the noise, such that the compression process effectively denoises the audio.

However, even with overlapping voices, Opus voice *still* meets or outperforms the other two codecs, with only a 0.07 greater WER than mp3 at 8 kbps, and 0.17 *less* WER than mp3 at 16 kbps. This is surprising, as one might expect the main features of the background noise to be kept by nature of it being speech, making the audio harder to transcribe.

It should be noted that “better” is relative here; at 8kbps, Opus voice nets an average WER of 0.51 with environmental noise (park noise) and 0.58 with overlapping voices (meeting noise). These are not great, especially compared with the 0.06 WER with no noise. Thus, while Opus voice does work better than the other codecs, it is still heavily affected by extraneous noise.

5.3 - Results by SNR

Doing one final split of the dataset, we can uncover the effect of the amount of noise on performance.



Keeping in mind that a higher SNR means less noise, it is clear the difference that noise makes. For park noise, with a SNR of 10, the best WER was 0.91 from Opus voice, meaning 91% of the output was errors. Going to an SNR of 20, the best WER dropped drastically to 0.06, also from Opus voice. The same happened for the speech with overlapping voices added (the meeting noise). This is easily attributed to the fact that more noise covers up the speech, so Whisper would perform worse no matter the compression method.

Some of these graphs are also surprising in that I expected the WER to generally decrease as bitrate increased. One might expect that the higher the bitrate, the better quality the sound no matter the codec, and Whisper would be able to perform better leading to a lower WER. However, some of these graphs plateau at points or jump up and down erratically, achieving lower WER at lower bitrates or drastically higher at higher ones. This might be due to the limited data used in this experiment. If more clips from more speakers were put through this same process, then a more reasonable and expected result might appear.

6 - Discussion and Limitations

At a first glance, these results clearly support the original hypothesis. Opus voice far outperformed mp3 and general-audio Opus (as well as downsampling) at the lowest bitrates. This means that, with Opus voice, one can compress speech at 8 kbps, transfer it, decompress the file, and use Whisper to transcribe said audio, and still get a lower overall word error rate than if the same process was done with mp3 and general Opus. This continues to be true even if the file was compressed with mp3 or general Opus at a *higher* bitrate, which would result in a larger file.

Additionally, even taking into account the different types of noises added on top of the speech, Opus voice still outperformed mp3 and general Opus. This is true even for the case of overlapping speech, where a codec designed to retain speech information would be expected to get worse performance. Instead, Opus voice performed close to or better than mp3 and general Opus at every bitrate tested. This difference only widens with lower SNRs, meaning with more noise in the way, Opus voice tends to be less hindered at lower bitrates. In fact, it seems to perform even better at the lower bitrates, even when compared to itself, as in the case for park noise at an SNR of 10 dB.

This brings up an interesting pattern in the results: sometimes a lower bitrate with the same compression method would lead to a lower WER. It is certainly unexpected that removing more information using the same method would lead to better intelligibility. This could be because “removing the unimportant parts” in speech audio roughly translates to “removing the non-speech parts”, even for the non-speech-specific codecs. This would mean having lower bitrates means throwing out more noise, so WER would increase. However, the pattern isn’t consistent enough to make any generalization. Thus, for now, it is safer to attribute this to the variation caused by the small amount of data used. More data and more hyperparameters would have to be tested in order to make a definite claim.

In addition to this, as mentioned before, these scores are all evaluated relatively. a 0.58 WER using Opus voice with overlapping voices, while better than mp3 and general Opus, is still not good by any means. For the user, that means when there are voices in the background (which is often), more than half of the words they are trying to transcribe will be incorrect. In other words, in this case, “best” does not really mean “good”. There may certainly be better codecs out there. For example, other speech codecs, like EVS, may be better for this specific task. More rigorous testing would be necessary to know for sure.

This brings to light an early challenge and glaring limitation of this project, which was the small amount of data I was restricted to. I was running Whisper locally on my laptop, meaning it took 30 seconds to a minute to simply load a roughly 10-second audio file, run it through Whisper, and evaluate the results. For 200 files, this took about an hour and a half, not including the time it took to add the noise, compress and decompress, etc. Due to how fast this project scales, adding speakers, speech files/noise files, or more values for hyperparameters like signal to noise ratio or bitrate proved to be a huge challenge. For a sense of scale, I originally planned on using 4 files each from 4 speakers, tested with 4 different types of noise at 4 different signal-to-noise ratios, compressed with 6 different codecs at 6 different bitrates. This would

result in 9,216 files and take 76.8 hours to fully evaluate, at least on my machine. Thus, I was restricted to a smaller set of data for this project.

This lack of data has led to a lot of variation in the final results, with the graphs jumping around erratically instead of showing any consistent pattern. It also means that the results don't fully take into account variability in voices. While two relatively different voices were selected to try and mitigate this effect, there is only so much variability that can be captured with two voices. While in general, the results are still interpretable, the lack of data definitely limits the generalizability of this project.

7 - Conclusion and Future Work

7.1 - Conclusion

While the raw results seem to point to Opus voice being the best codec, they are far too erratic and unreliable to draw a definite conclusion. However, what these results do show is that the method of compression can affect the performance of an ASR model like Whisper in a significant way. In the worst cases, Whisper performed almost three times worse on speech compressed with general Opus when compared to mp3 or Opus voice. While these differences may be exaggerated due to the small sample size, they are still significant enough to show that this topic requires further inquiry, especially for cases such as automatic transcription in courtrooms, where a small increase in error could have enormous consequences.

7.2 - Future Work

While the original question could not be answered in a satisfactory manner, there is a clear direction to go with this project. In the future, more data would have to be used, and more hyperparameter values would have to be tested. In order to fully gauge theis effectiveness at different levels of noise, for example, many different SNRs would have to be tested. Different codecs might also be tested. While Opus and Opus voice have outperformed other codecs in blind listening tests (Hoene et. al., 2013), that only means humans perceive them as better. This is because psychoacoustic models are baked into audio codecs so they preserve what's important for humans. We as a species are not sure if neural ASR models approximate human hearing to transcribe text. If they do not, then another codec that might sound worse to humans could be the key to maximizing ASR model performance while minimizing file size.

Overall, this project demonstrates that there are limitations in ASR to be considered before the model even sees the data. As the worlds of compression and machine learning develop, the intersection between these two fields becomes more than ever an important place to explore.

Works Cited

Hoene, C., Valin, J.-M., Vos, K., & Skoglund, J. (2013, May 17). *Summary of Opus Listening Test Results.* IETF Datatracker. https://datatracker.ietf.org/doc/html/draft-ietf-codec-results-03

Marangon, J. D. (2023, November 16). *How to Calculate the Word Error Rate in Python.* Medium. https://medium.com/@johnidouglasmarangon/how-to-calculate-the-word-error-rate-in-python-ce0751a46052

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). *Librispeech: An ASR corpus based on public domain audio books.* 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). https://doi.org/10.1109/icassp.2015.7178964