Assignment 4: Text-to-Speech Evaluation

Emma Rafkin
Georgetown University
epr41@georgetown.edu

Abstract

In this assignment I explore the codebook size to compression ratio of Meta's EnCodec model as well as the quality improvement for different types of audio files as the number of codebooks increase. Then I explore six different text to speech models to identify the best one to integrate into my final chatbot.
Code for this project is hosted on Github[1].

1 Introduction

Speech is the easiest form of communication for many people. Realistic text to speech (TTS) generation is an important feature in creating useful AI agents. However, audio can be very expensive to generate and store. The task of digitizing a continuous audio stream and compressing it to be evaluated by a model or stored and then decoding that compression into the same form with little to no quality degradation is extremely complex. This paper explores how neural codecs balance compression and quality, as well as tests which TTS systems provide the best quality-latency trade-off.

TTS systems originated as concatenative systems, in which recorded speech was added together to form the final sound. Then statistical models like Hidden Markov Models were used to predict acoustic parameters which were passed into vocoders to generate speech. Now we are in the era of neural TTS systems. These involve an encoder to take the

2 Methods

2.1 Hardware Specifications

I ran phase 1 on a Mac Intel Chip because the model used for this task did not require a GPU. I ran Phase 2 on a Ubuntu machine with 125GB of RAM with 8 cores of CPU.

[1] https://github.com/erafkin/ling4467_assignment_1

2.2 Part 1: Neural Codec Evaluation

I used Meta's EnCodec (Défossez et al., 2022) model because it was free and easy to use. Initially, I attempted to use Soundstream but it appeared that the version that I had downloaded only used 2 codebooks, which led to poor decompression results and nullified the purpose of this experiment. The model version I used required the original audio be 24 kHz, which I thought was a reasonable quality expectation.

I tested the model on 10 audio files, 9 of which were downloaded from Freesound[2]. These files were all licensed to be downloaded, but not distributed. I downloaded a female speech sample, a male speech sample, and a poetry sample. I also used one of my audio recordings from an earlier assignment. For non-speech audio, I used two bird sounds, a clip of piano, a clip of guitar, a car crash sound effect, and a movie celebration score. All of these clips varied in length but were all under 30 seconds.

I compressed each clip using EnCodec and decompressed it with 1-32 codebooks (32 is the maximum number for the model). I then listened to each decompression and noted its quality. I also saved the compression and bitrate statistics for each original audio file.

2.3 Part 2: TTS System Evaluation

I tested 6 TTS systems, 5 of which are available on Huggingface and were run locally, as well as the Google Translate TTS (gTTS), which is an API service. The Huggingface pipeline object will automatically load and run models that have the right configuration, however some models are either deprecated or not build to this specification. I unsuccessfully tried FastSpeech2Conformer (Guo et al., 2020), VibeVoice (Peng et al., 2025), and Chatterbox (Resemble AI, 2025). In the end, I ran

[2] https://freesound.org/

TTS using gTTS, Microsoft's Speecht5 (Ao et al., 2022), Meta's VITS (Pratap et al., 2023), Parler (Lyth and King, 2024), and Bark[3] small and large.

Since I had a GPU at my disposal, all of these models were simple to run. Without this machine, I would have to rely on more API services, which would have cost money once any free trial runs out. The simplicity of using Huggingface for all 5 local models was a large draw, because even though each one required a few tweaks, the pipeline wrapper handled a lot of the code necessary to run the models.

I passed through 15 test prompts:

1. Hello world, my name is Emma Rafkin.
2. I am ready to eat dinner.
3. It is cold outside today
4. He likes to read, but he read the book that she is reading already
5. She blew out a breath, and the cold air made the blown breath appear blue
6. That was a totally crazy thing for you to do!
7. Why do you think that?
8. Give papa a cup of proper coffee in a copper coffee cup.
9. Peter Piper picked a peck of picked peppers
10. Fuzzy wuzzy was a bear. Fuzzy wuzzy had no hair. Fuzzy wuzzy wasn't fuzzy, was he?
11. Some people think that 74.5 degrees Fahrenheit is hot.
12. In 1986, the USA got 1st and 2nd place at the Olympics
13. The algorithm uses stochastic gradient descent for optimization.
14. What are you eating?
15. You are eating what?

These prompts were created to test the systems on emotion, tongue twisters, homonyms, prosody, and generation of varying length. I ran each test prompt through each model and measured the time it took to run the model. For some of the systems this also included loading up the model and none of the inference was batched. While this might be more efficient in the final chatbot environment, there was an equal latency comparison across models.

[3] https://github.com/suno-ai/bark

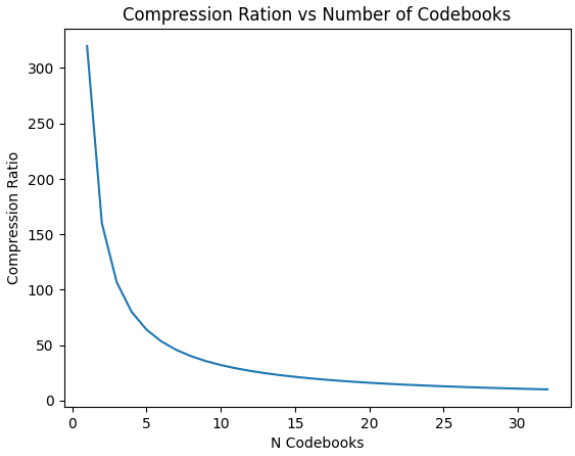

Figure 1: Compression ratio vs number of codebooks. After around 7 codebooks the tradeoff degrades rapidly.

After each file was created, pairs were created of each model's synthesis of the prompt and played back for an evaluator (me). I would then select which synthesis I thought was better, allowing for the answer to be neutral. I first graded based on completion of the prompt (getting all the words and no insertions), then on pronunciation, then on humanness.

3 Results

3.1 Part 1: Codec Results

Figure 1 demonstrates the tradeoff between number of codebooks and compression ratios. There is a "knee" in the curve right after 5 codebooks. However, this does not necessarily align with the audio quality. The human speech sounds robotic with the fewest amount of codebooks, and remains relatively "fuzzy" until around 18 codebooks. Depending on the type of non-speech audio fewer or more codebooks can be tolerated. Bird whistles needed many codebooks, while the car crash was reproduced easily by around 8 codebooks. The music depended on the type of music. Piano and guitar needed a lot of codebooks (around 24) to not sound muffled, while the "celebration" score only needed around 10 to sound fine to my ear.

Despite the high success rate of the EnCodec model, there remained some failure cases. The high pitched tweets of the robin were never fully decoded properly. This could even have come

Model	A /B score
Bark Large	47
Speech T5	42
GTTS	36
Bark Small	35
Parler	28
VITS	10

Table 1: Overall TTS model scores from A/B testing

from the resampling into 24kHz. Regardless of whether it initiated from the model or the resampling, running it through the process caused the bird sounds to be quite “robotic". This was not the case for the lower owl hoots. Overall, the more codebooks led to higher quality decoding, but the full 32 were never really needed. Most of the audio files achieved good sounding results with around 24 codebooks.

3.2 Part 2: TTS Results

Overall scores from the A/B testing can be seen in Table 1. Generation latency for the models across the prompts can be seen in Figure 2.

For this task, it was unsurprisingly the case that the best model was also one of the slowest. Unlike other tasks, however, in this case quality was not directly mapped to latency. Parler took nearly as much time as Bark large and it performed second to worst. gTTS is not necessarily a fair comparison because this model was run on the API– however performed very well and took essentially no time to run. The longer latency prompts were also the longest prompts, which is not surprising because the models have to attend to more when the prompts are longer.

The shorter declarative sentences were the easiest across all the models. Systems differed the most with the prompt that included the temperature “74.5 degrees". Both of the Bark models and gTTS were able to say this number, while none of the others could seem to produce it. This was the same for the year “1986". An interesting artifact for that particular prompt was that the Parler model started speaking a language that sounded a lot like German, but ended in English after the numbers were over. These experiments show that numbers are particularly difficult for TTS systems.

The tongue twisters led some models into chaos. A notable case was that the Bark small model generated (eerie) singing for the Fuzzy Wuzzy poem. This also happened for that model for the “she blew out a breath" prompt. The VITS model really struggled with the tongue twisters, generated mumbled/garbled speech which was also in a non-American accent. Bark Large, which usually did well, generated silence for the “give papa" tongue twister. Another failure case was the prompt with my name. Mainly, some of the models failed to pronounce my last time correctly–but proper nouns are particularly tricky.

It sounded like Bark Large was trained on a lot of news and radio data, as the voice was often a "newscaster" voice. This means that the speech was generated with a particular newscaster cadence. This could be seen especially in the prompts about the weather or about the Olympics. This was the most human-sounding model that did not struggle at all with numbers or hard words. Even though gTTS sounded robotic, it was the most consistent with the challenging prompts. It did not skip any words and did not add any artifacts. The VITS model specifically struggled with the tongue twisters and with a lot of the pronunciations, oftentimes melding together words.

4 Discussion

After the exploration of the EnCodec model in phase 1, it seems that for speech the optimal compression rate is around 20 codebooks to maintain high quality audio. However, for the audio to be intelligible, much fewer codebooks are needed. The speech was still easily intelligible with around 5 codebooks, but there was a lot of feedback and noise in the decoded audio. For music or nature sounds more codebooks are needed, and for non-speech sound effects fewer codebooks are needed. The quality of the compression can really be seen as an upper bound for neural TTS, because the better the neural codec, the more human-sounding and robust the TTS output is. My final ranking of the systems are:

1. gTTS
2. Bark Large
3. Speech T5
4. Parler
5. Bark Small
6. VITS

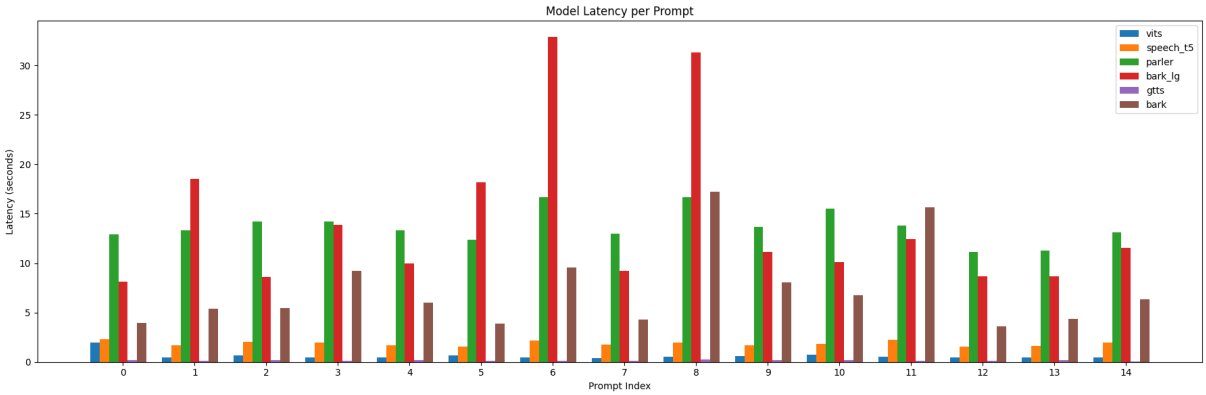

Figure 2: Latency for each TTS model for each prompt. Bark Large and Parler are consistently the slowest while gTTS is almost immediate.

I think that gTTS was the best despite it's robotic voice. The inhuman quality does not bother me that much and its lack of hallucinations and speed are important factors that the other model's can't guarantee. Bark Large is ranked second and Speech T5 third due to their A/B test results. I bumped Parler higher on the ranking because despite its latency, the fact that Bark Small would generate eerie songs sometimes makes me want to integrate it less. I think that having the chatbot burst into a spooky robotic song would be disarming for a human user. Finally, since VITS produced garbled speech for a lot of the more difficult prompts, it ranks last on my list.

All of these models were free so cost is not an issue, however, hardware certainly is. Some of these models took over 30 seconds to generate speech on a GPU, which is not feasible for the end use case of the chatbot. Conversational AI requires fast speech production because humans are impatient. Therefore, it is worth taking a hit in quality in order to lower latency of the models. Additionally, my personal laptop where I plan to be running my chatbot does not have a GPU, therefore a locally intensive model is not a good choice to run for my system.

5 Integration with Chatbot

I ended up integrating the gTTS system into my personal chatbot. Latency was the main reason why I did this. While I am certainly sacrificing my chatbot's humanness, the speed and accuracy of pronunciation that gTTS provides outweighs the benefits of using other models like Bark Large. Especially since I am running this on a CPU-only device, a large model is not feasible to run. Other than the robotic voice, the main drawback with using gTTS is the fact that it does not have inherent playback capabilities. Unfortunately, due to how the library is set up, I have to save the output to a file and call my playback method that I wrote in Assignment 2. This is not a huge issue, but is slightly annoying.

The speed of the overall system does not increase very much after integrating gTTS (which is why I chose it). The main bottlenecks are running Whisper and waiting for the LLM to respond. For a question that generates a long answer (e.g. who is the governor of Illinois), the whole system can take around 30 seconds to run. But for a shorter prompt ("it's cold outside today"), it only takes around 10 seconds.

After integrating gTTS with my system I found that another bonus of choosing the highest-accuracy model is that it is even able to "read" emojis. My LLM (GPT-OSS 120b) responds with many emojis and the TTS model was able to handle them fine. Particularly, on the prompt about it being cold, the LLM put a snowflake emoji in the response and this did not trip up the TTS model at all.

6 Conclusion

Neural codecs can encode and decode audio files to an impressive degree. Meta's EnCodec model can compress files up to 300% with just 1 codebook. However, doing so leads to extremely poor decoding. Therefore, it is important to use at minimum around 8 codecs for consistent decoding. Although the model has 32 codebooks available, many audio files could achieve decent quality using only around 24 codebooks.

TTS systems vary in performance. Some are quite human-like but require a lot of computational power. Some of the more powerful TTS systems also hallucinate or trip up over common artifacts such as numbers. At the end of the day, a more consistent and fast, yet robotic sounding model was the best choice for the deployment scenario of a chatbot running on a CPU. If for some reason it is desired that the model sound more “human", then a larger model such as Bark large would be optimal for that task.

Different deployment scenarios would require more evaluation. Speech synthesis evaluation can be done quantitatively, but also should be done as it was in this paper: qualitatively. Even though it can be tedious to listen to so many combinations of audio files, the models vary so much in their production and in their pitfalls that it requires a human evaluation to understand which will be optimal for the specific use case.

One limitation of my codec model choice is that the main quality degradation of my sounds potentially initiated from the resampling of the audio. This means that it is not the fault of the model and might have been mitigated by using a model that worked with 48kHz audio files. Another limitation of my study is that I only explored one API-based model because wanted to take advantage of the simplicity of using the Huggingface Library. Likely there are other API-based models that are as fast as gTTS but do not have the same robotic quality to the voice.

References

Junyi Ao, Rui Wang, Long Zhou, Chengyi Wang, Shuo Ren, Yu Wu, Shujie Liu, Tom Ko, Qing Li, Yu Zhang, Zhihua Wei, Yao Qian, Jinyu Li, and Furu Wei. 2022. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. *Preprint*, arXiv:2110.07205.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. 2022. High fidelity neural audio compression. *Preprint*, arXiv:2210.13438.

Pengcheng Guo, Florian Boyer, Xuankai Chang, Tomoki Hayashi, Yosuke Higuchi, Hirofumi Inaguma, Naoyuki Kamo, Chenda Li, Daniel Garcia-Romero, Jiatong Shi, Jing Shi, Shinji Watanabe, Kun Wei, Wangyou Zhang, and Yuekai Zhang. 2020. Recent developments on espnet toolkit boosted by conformer. *Preprint*, arXiv:2010.13956.

Dan Lyth and Simon King. 2024. Natural language guidance of high-fidelity text-to-speech with synthetic annotations. *Preprint*, arXiv:2402.01912.

Zhiliang Peng, Jianwei Yu, Wenhui Wang, Yaoyao Chang, Yutao Sun, Li Dong, Yi Zhu, Weijiang Xu, Hangbo Bao, Zehua Wang, Shaohan Huang, Yan Xia, and Furu Wei. 2025. Vibevoice technical report. *Preprint*, arXiv:2508.19205.

Vineel Pratap, Andros Tjandra, Bowen Shi, Paden Tomasello, Arun Babu, Sayani Kundu, Ali Elkahky, Zhaoheng Ni, Apoorv Vyas, Maryam Fazel-Zarandi, Alexei Baevski, Yossi Adi, Xiaohui Zhang, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. 2023. Scaling speech technology to 1,000+ languages. *arXiv*.

Resemble AI. 2025. Chatterbox-TTS. https://github.com/resemble-ai/chatterbox. GitHub repository.