1: Related Work
---
1. Choose at least 2 pieces of related work per group member. For each piece of related work, write a paragraph that includes:
   - Summary of the main contributions of that work.
   - How your proposed project builds upon or otherwise relates to that work.

Wav2Vec
Published in 2019, Unsupervised Pretraining for Speech Recognition introduced wav2vec, a neural network architecture designed to learn representations for audio that could help improve acoustic model training. Wav2vec uses a simple multilayer CNN architecture that takes raw audio as input and computes a general representation that can be used as input to a speech recognition system. The network is trained by using a contrastive loss between the future predicted audio and the true future audio sample. Their main contribution was being able to produce robust audio embeddings that could achieve state-of-the-art word error rates of around 2.4% on the nov92 dataset at inference time while only using a fraction of the labeled dataset required.
Citation: Schneider, Steffen, et al. "wav2vec: Unsupervised pre-training for speech recognition." arXiv preprint arXiv:1904.05862 (2019).

Since we would like to analyze audio for the purposes of speaker diarization, wav2vec could provide a useful method of embedding audio in different segments of a signal. We can then cluster the embeddings to predict when a unique person is speaking.

Wav2Vec 2.0
Wav2Vec2 attempts to further improve robustness of audio representations of wav2vec. The algorithm encodes speech audio via a multi-layer convolutional neural network and then masks portions of the resulting latent speech representations. The latent representations are fed to a transformer network to build contextualized representations and the model is trained via a contrastive loss where the true latent space values are to be distinguished from distractors. Again, wav2vec2 was able to perform well for word error rate when pre-training on unlabeled data for speech processing: when using only 10 minutes of labeled training data, or 48 recordings of 12.5 seconds on average. They achieved a WER of 4.8/8.2 on test-clean/other of Librispeech. Librispeech is a voice dataset containing 1000 hours of spoken English.

Citation: Zhang, Aonan, et al. "Fully supervised speaker diarization." ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019.

We can use wav2vec2 for the same purposes as wav2vec: for creating a descriptive embedding of audio for clustering and comparison. In our benchmarking, wav2vec2 had a slightly lower diarization error rate of 0.56 as opposed to the 0.7 error rate of the traditional wav2vec. However, wav2vec2 had a longer average inference time of about 7 seconds as opposed to wav2vec's 3 seconds.

HuBERT

Applying BERT-style representation learning to audio tasks presents a few problems not present in text. One primary issue is that there is no segmentation inherent in the signal. HuBERT uses a clustering approach over outputs from a CNN-based backbone (similar to wav2vec approaches). This provides tokens and labels which can be masked and learned in a BERT style. From this, HuBERT learns a joint audio and language model. Similar to other transformers, this model is relatively large compared to our other baseline models at 92M parameters. The authors also train large and extra large versions at 317M and 964M parameters, respectively. These sizes are prohibitively large for use on device, especially when some existing models can provide reasonable performance at a much smaller parameter count.

Citation: Hsu, Wei-Ning, et al. "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units." arXiv preprint arXiv:2106.07447 (2021).

Resemblyzer

Resemblyzer is a voice encoder similar to wav2vec and wav2vec 2.0 and is based on the paper, Generalized end-to-end loss for speaker verification. In Generalized end-to-end loss for speaker verification, the authors perform a process called enrollment in which a speaker provides a few utterances which are then used to estimate a speaker model. The vector embeddings of these utterances are averaged to build that speaker model. For verification, a tuple consisting of one evaluation utterance, x, and the enrollment utterances, M, are passed into an LSTM. The network uses both a softmax and contrastive loss to both push the embedding vectors toward their own centroids, and pull them away from the other centroids. The centroids being the enrollment or reference utterances of the various speakers. Another one of the contributions was in implementing the MultiReader to combine different data sources, enabling the models to support multiple keywords and multiple languages. Their model was able to have an Equal Error Rate of 3.55%. EER measures the error rate between the true and predicted utterrer of an utterance.

Citation: Wan, Li, et al. "Generalized end-to-end loss for speaker verification." 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018.

Resemblyzer contains a demo in their repository that could perform both embedding, clustering, and diarization all in a single pipeline. We may choose to focus on this library since it contains an end to end framework from audio file to diarization inference all in a single pipeline. We would need to include this pipeline as part of our larger project where we can try to pair the model with some hardware or try to make some decisions based on the inference.

UIS RNN

Another potential solution to the speaker diarization problem or "who spoke when" was introduced in 2019 in a joint effort between Google and Columbia University in their paper, Fully Supervised Speaker Diarization. In the paper, the authors introduce unbounded interleaved-state recurrent neural networks (UIS-RNN). The RNN accepts speaker-discriminative embeddings (a.k.a. d-vectors) from input utterances and each individual speaker is modeled by a parameter-sharing RNN. Recent work involving audio embeddings has been shown to improve diarization performance replacing i-vectors with neural network embeddings, a.k.a. d-vectors. These d-vectors are more representative and robust since the networks can be trained with big datasets accounting for varying speaker accents and acoustic conditions. UIS RNN was fully supervised and trained on examples where time-stamped speaker labels are annotated. They achieved 7.6% diarization error rate on NIST SRE 2000 CALLHOME, which is better than the state-of-the-art method using spectral clustering. Another contribution of USI RNN is that it was capable of decoding/clustering embedding in an online fashion compared to most state-of-the-art systems which relied on offline clustering.

Citation: Zhang, Aonan, et al. "Fully supervised speaker diarization." ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019.

Unlike wav2vec, UIS RNN uses a diarization inference algorithm. It accepts d-vectors (i.e. audio embeddings) and then using the RNN, it clusters them into what it thinks are separate speakers. If we choose not to use the integrated demo that Resemblyzer uses, we can pair one of the embedding models like wav2vec with the clustering/diarization that UIS RNN provides.

Speaker Diarization with GNN

Speaker Diarization with Session Level Speaker Embedding Refinement Using Graph Neural Networks (Wang et. al, 2020) proposes using GNNs to produce more separable embeddings for each speaker segment. Using a pretrained backend model to produce initial embeddings, this work then learns a mapping from each embedding to a graph, on top of which a number of graph neural network layers are used to produce refined embeddings. Affinity propagation is used to cluster these embeddings and produce an affinity matrix, which can then be used to assign identity to each segment. Attaching numerous GNN layers on top of an already expensive backend model is an expensive approach in comparison to other end-to-end or light-weight frontend approaches. Thus, we elected to use approaches that brought down to size of the frontend, like clustering directly off backend embeddings, or using a lightweight RNN or LSTM on top of a backend.

Citation: Wang, Jixuan, et al. "Speaker diarization with session-level speaker embedding refinement using graph neural networks." ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020.

2: Baselines

---

1. What are the baselines that you will be running for your approach? Please be specific: data, splits, models and model variants, any other relevant information.

We attempted to use wav2vec, wav2vec 2.0, modified_cpc, vq_wav2vec, wav2vec_xlsr, and Resemblyzer.

All of the models are pretrained. For testing, however, we are using the VOX Converse Dataset. It contains 10 wav files of snippets from various news broadcasts. In each file, there are at most 4 unique speakers.

We load each of the models and then pass a downsampled version of the wav files into the model for embedding. We downsample each traditional 44 KHz wav file into having approximately 1 KHz sample rate. Without the downsampling, embedding the models took well over an hour each time.

2. These baselines should be able to run on your device within a reasonable amount of time. If you haven't yet tried to run them, please include a back-of-the-envelope calculation of why you think they will fit in memory. If the baselines will not fit in memory, return to (1) and adjust accordingly.

3. How will you be evaluating your baselines?

We are evaluating our baselines using the diarization error rate. Since speaker diarization attempts to segment an input audio stream by speaker identity, the DER is a measure as the fraction of time that is not attributed correctly to a speaker or to non-speech. This will be compared along with inference time and model size.

In other words, given an input audio signal, our networks can create 512-element vector embeddings for each 10 ms stride. Once the audio is converted into a list of embeddings, we attempt to find 4 clusters in the embeddings using scikit-learn’s K-Means clustering algorithm. We assign each vector one of the cluster categories and then compute the diarization error rate. This clustering approach is very common to speaker diarization algorithms, with many advances simply implemented further processing on the backend embeddings before performing some form of clustering.

4. Implement and run the baselines. Document any challenges you run into here, and how you solve them or plan to solve them.

Major difficulties included downloading and setting up s3prl library on-device. Despite its promise of being a unified front-end framework that provides access to several audio embedding networks, s3prl is very poorly optimized for non-Intel systems as we discovered in the previous lab. The ARM environment required for s3prl to run is also very fragile and we encountered numerous problems

We also ran into an error where our system kept running out of memory. This was caused by processes containing our model testing code that weren't killed when we exited the device or when the internet connection was dropped. Also, the default sampling rate resulted in massive tensors and tons of compute and memory required to cluster.

The table below shows diarization error for each sample wav file, along with some efficiency statistics.

Model Name	**WAV 1**	**WAV 2**	**WAV 3**	**WAV 4**	**WAV 5**	**Avg. DER**	**Avg. Inference Time (s)**	**Model Size (Params)**
Resemblyser	0.74	0.70	0.66	0.53	0.53	0.63	11.54	1,423,616
vq_wav2vec	0.57	0.66	0.39	0.46	0.35	0.49	9.12	34,145,408
wav2vec2	0.64	0.58	0.63	0.53	0.44	0.56	9.79	95,044,608
modified_cpc	0.80	0.77	0.79	0.82	0.87	0.81	2.65	1,843,456
wav2vec	0.84	0.60	0.78	0.61	0.66	0.70	8.64	32,537,088
HUbert	0.57	0.73	0.40	0.64	0.43	0.55	9.90	94,697,600

5. If you finish running and evaluating your baselines, did the results align with your hypotheses? Are there any changes or focusing that you can do for your project based on insights from these results?

These results did not align with our hypotheses. All the backend models except Resemblyser are not fine tuned on the diarization task. Thus, we expected these backend embeddings to not perform as well as Resemblyser. However, a number of the wav2vec-based models beat Resemblyser in average DER at a lower inference time. The inference time comparison is due to Resemblyser being sequential (an LSTM), while the wav2vec based models are transformers, and thus operate in parallel over the input audio.

Additionally, we found that the frequency of the sampled audio truly dominated runtime. It may be interesting to look into performance and runtime trade offs and different levels of subsampling on the nano. This sampling frequency presents a significant bottleneck on the nano, so dedicating a part of the final report to this aspect may be fruitful, and isn't addressed by the previous works which assume an adequate sampling rate.

All of the current baselines that we implemented cluster the upstream audio segment embeddings into a fixed number of speaker-groups. This restricts us from handling the scenario with overlapping voices in a segment. We may also want to look at possible (on-device) extensions of the UIS paper since it doesn't require the number of speakers to be mentioned beforehand and can also benefit from supervised training data (as opposed to unsupervised clustering).

3: Extra

More related work, more baselines, more implementation, more analysis. Work on your project.

We also made a recording of each member saying the phrase "A quick brown fox jumps over the lazy dog." The recording is around 9 seconds long, with HuBERT giving the most reasonable diarization results.

FAQ
---
1. Our baseline is the SotA model from XXX 2021 which doesn't fit on device.

Yikes! I think you meant to say -- "We used some very minimal off the shelf components from torchvision/huggingface/... to ensure our basic pipeline can run"

2. We're evaluating our baseline on accuracy only

I think you meant to say -- "We plan to plot accuracy against XXXX to see how compute and performance trade-off. Specifically, we can shrink our baseline by doing YYYY"