CS230

Creating a Robust Audio Representation via Self-Supervised Contrastive Learning for Speaker Diarization

Priya Khandelwal and Neel Narayan
Department of Computer Science
Stanford University
priyak9@stanford.edu and neelsn@stanford.edu
Code: https://github.com/neelsnarayan/cs230_final_project

Abstract

Self-supervised contrastive learning is a technique that encourages different augmentations of the same input to have more similar latent representations, and distinct inputs to be very far apart in the embedding space. Thus, contrastive learning seems to be a promising way to train a model to generate acoustic embeddings for speaker diarization: the task of identifying which segments of an audio clip containing multiple speakers were spoken by the same person - a task that is usually done by clustering embeddings. In this paper, we explore the effectiveness of this novel framework by training and testing a SimCLR-inspired self-supervised model on the Flickr Corpus dataset after generating multiple augmentations, or "views," of different single-speaker audios. We compare our model's diarization error rate (DER) against LSTM d-vector baselines and existing works for speaker diarization. Our model outperformed the baselines, achieving a DER of 0.189, indicating that with a more complex clustering approach, this architecture is expressive enough to generate competitive embeddings for speaker diarization. We conclude that with additional improvements, this method can achieve even more robust performance.

1 Introduction

AI tools like Amazon Alexa, Google Home, and IBM Watson make an attempt to identify the speaker in question when processing audio–also known as automatic speaker diarization–but still lag far behind human proficiency.[1][2]

After large audio chunks are broken down into single speaker segments, SOTA (state of the art) speech diarization workflows typically take two main steps: translating the segments to some latent embedding and then clustering them in the embedding space to distinguish between speakers.[2]

Contrastive learning is a self-supervised technique that encourages different augmentations of the same input to have more similar latent representations, and distinct inputs to be very far apart in the embedding space. For vision-language models and variational autoencoders, contrastive learning has proven to be highly successful in learning robust representations of images. However, its use in generating representations for audio data is largely unexplored.

In our project, we train a contrastive learning method on audio data to discover a more robust and granular representation for acoustic information, allowing for better clustering of audio embeddings for downstream tasks, particularly speaker diarization.

2 Related Work

A traditional ML-based approach to generating latent representations for audio is i-vectors, which are created after applying a dimensionality reduction technique to the audio data that has been embedded via a GMM model. However, i-vectors make the assumption that the audio data follows a Gaussian distribution. d-vectors, on the other hand, average the activations in the last hidden layer of the RNN's, time-delayed neural networks, or transformers.[3][4][5].

However, both of these approaches heavily rely on labeled data and do not actively prioritize augmentation invariance.

Contrastive learning has recently been applied to language-agnostic emotion recognition in audio data and end-to-end speech translation. Google Cloud has also recently added speech-to-text API that incorporates successful diarization capabilities trained on an unsupervised framework. Given this

,

promising work, we think a self-supervised approach can be applied to speech diarization.[6][7].

3 Dataset

The dataset we use for training, validation, and testing is the Flickr 8k Audio Caption Corpus. This corpus contains 40,000 english-spoken captions of 8,000 images narrated by 183 unique male and female speakers. The audio clips range from 3-7 seconds each, with variance in volume, speed, enunciation, and accent, thereby allowing for more robust training. Sample expressions from the audio files contain captions such as "two children are playing ice hockey on frozen ground outside" and "a man rides a bicycle on a rocky trail along a large river." The data is sampled at 16000 Hz with 16-bit depth, and is stored in the Microsoft WAVE audio format. We use an 80%-10%-10% split.

For additional testing, we used the JL Corpus, a dataset of 4 male and female speakers speaking 2400 English phrases in 7 different emotions that we presume closely follows the distribution of the data we used for training.

3.1 Baseline Preprocessing

All of our data preprocessing for the baselines is done in preprocessing.py. With the Flickr Corpus, we were provided with a text file that mapped each .wav to the index of a specific speaker. From this, we could leverage the 'who said what' mappings. To model the human hearing property at the feature extraction stage, we decided to use the Mel scale to map the audio data to the frequency that mimics what humans would perceive (humans perceive changes in low frequency sounds with more nuance than they do changes in high frequency sounds). We extract 40 Mel Frequency Cepestral Coefficients (MFCCs) from each audio sample using the python Librosa package and store them in a .npy file, and then save them all in our train_features folder. In the event that the .wav files are given to us in a different format, we wrote a program (create_text_file.py) to generate a space delimited text file with each .wav representation corresponding to the speaker present in that audio sample. We used this approach on the additional JL corpus test set as well.

3.2 Contrastive Data Augmentations

In contrastive learning, the choice of the data augmentation technique is the most crucial hyperparameter since it directly affects how the latent space is structured and influences the view-agnostic robustness of the model. In particular, we want to choose data augmentations that still preserve speaker identity for the same input. Since our contrastive learning approach is primarily vision-based (explained further in the methods section), we decided to feed spectrograms into the model instead of MFCCs. Our novel insight is that if we generate acoustic augmentations to the raw audio before generating spectrograms, and then visual augmentations after generating spectrograms, we obtain more meaningful views for the contrastive method to train on and learn more generalizable features.

The transformation that we chose to apply to our samples was a time stretch implemented in torchaudio, which we applied by randomly speeding up/slowing down parts of the audio. This highlights certain voice features such as pitch and vocal timbre, which are unique to speakers, but don't fundamentally alter the audio.

After applying the time stretch contrastive data augmentation technique, we generate spectrograms of each audio sample and choose visual augmentations based on existing literature. On our spectrograms, we randomly crop and resize the images, allowing us to distinguish between two situations: having a local view of the spectrogram, and having a neighboring view. In hyperparameter tuning, we found that relatively large crops made the most sense, as zooming into a small region of a spectrogram removes insight about its amplitude, frequency, etc.

Finally, we use torchvision to apply three other transformations on the already cropped and resized spectrogram: slight color distortion by changing the brightness, contrast, saturation, and hue, random grayscale conversion, and convolutional gaussian blur. In total, we generate 6 distinct random views of each image, and the dataloader chooses 2 at random from those 6 during each positive training step.

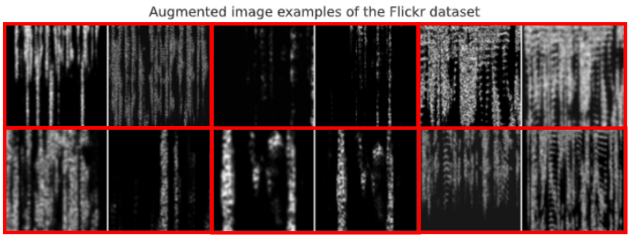

Figure 1: Sample Data Augmentations

Without these three additional transformations, the model might end up focusing only on the color histograms of the images and ignore other more generalizable features. By distorting the spectrograms and generating different views, the model cannot rely on this simple feature anymore. We've now generated spectrogram views that are distinct but that are easily distinguishable from speaker to speaker. Examples are shown in **Figure 1**: there are two randomly selected augmentations of the same speaker for 6 distinct speakers, as selected by the dataloader.

4 Methods

4.1 Baseline: d-vector LSTM

Though our initial plan was to have two baselines, a d-vector used jointly with manually extracted linguistic features, and LSTM, we instead decided to make one baseline with a d-vector of an RNN and try treating both the ReLU dense layer

Data Augmentation		
Hyperparameter	Description	Value
Time Stretch	stretch short-time fourier transform in time without modifying pitch	n_freq = 401, fixed_rate = random(0.5, 1.5)
Random Resized Crop	randomly crop and set image to specified size	size = 128
Color Jitter	randomly change the brightness, contrast, saturation and hue of image with given probability	brightness = contrast = saturation = .5, hue = .1, prob = .8
Random Grayscale	randomly gray image with a specified probability	prob = .2
Gaussian Blur	blurs image with randomly chosen Gaussian blur	kernel size = 9
Contrastive Model		
Hyperparameter	Description	Value
Batch Size	number of samples per propagation	size = 256
Hidden Dimensions	size of hidden layer	num_dimensions = 128
Temperature	factor to divide logits by to control randomness of predictions	temp = .07
Weight Decay	penalty to add to cost function to shrink weights	decay = 1e-4
Max Epochs	number of passes through neural network	epochs = 500

Table 1: Hyperparameters

and the Softmax dense layer as embeddings. In particular, we trained an LSTM on a speaker classification task and tried each of the aforementioned layers as a latent embedding for the downstream speaker diarization/clustering task [4].

The architecture of the seqential layers is as follows: there are two LSTM cells with a hidden unit size of 64, a dense layer of 350 units with ReLU activation and dropout (with 0.3 drop probability), and a dense layer of 500 units with a softmax activation. With the dense layers, there are two important considerations: (1) we add 350/500 units to the layer despite there only being 183 unique speakers in our data to guarantee separation and ease of distinction between each speaker (i.e. sparsity) and (2) we use softmax as our activation function because we are then implicitly clustering our speakers.

We implemented our RNN with Keras and Tensorflow. After doing a grid search on different values for the following hyperparameters, we ultimately found best performance to be: Adam Optimization with a learning rate of .0001, batch sizes of 32, and 500 epochs. We use sparse categorical crossentropy loss as our loss function and optimize on accuracy as our metric. Though the model metrics are not relevant to the clustering and diarization process, we report them as such: the model achieves a 98.89% accuracy on the val dataset, and though there may be some overfitting here, the model achieves a 95.44% accuracy on the test dataset, so we assume it is safe.

4.2 Contrastive Learning

Our main approach achieves a contrastive objective rather than a traditional predictive objective for generating embeddings. The loss function for a positive pair of examples is as follows:

$$L_{i,j} = -\log \frac{\exp\left(sim(z_i, z_j)/\tau\right)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp\left(sim(z_i, z_k)/\tau\right)}$$

Here, sim is defined as the cosine similarity between two vectors, which is computed after normalizing all the features in the L2 sphere. The indicator function in the denominator is used to capture only the pairs that contain the vector of interest, and the numerator contains positive pairs with the vector of interest. This loss is quite similar to cross-entropy or InfoNCE.

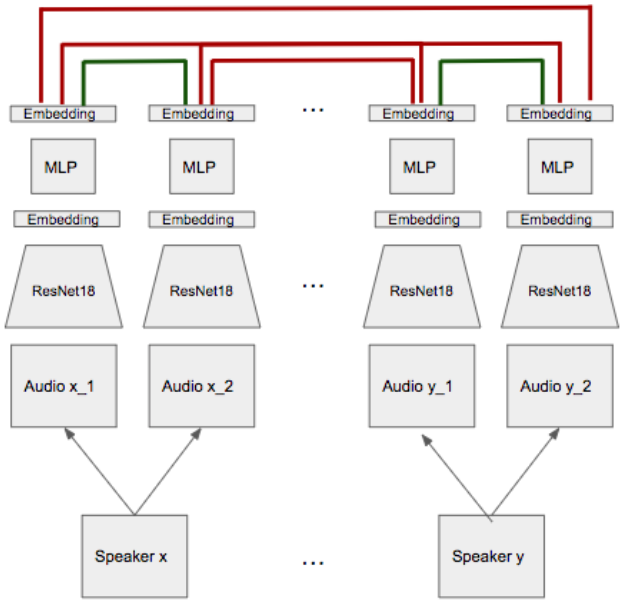

Figure 2: Self Supervised Model Architecture

Inspired by SimCLR, our self-supervised model architecture is as shown in **Figure 2**. We have a CNN encoder (ResNet18) followed by an MLP (the FC layer of the CNN, then ReLU/Softmax, and Linear layer) to generate a vector representation of two audio clips from the same speaker. In one version of our model, the last layer of the CNN uses ReLU, and in another, we try softmax. We are training the model to maximize agreement of same-speaker representations (green), and vice versa for different speakers (red), and use the according losses.

We implemented our contrastive models with PyTorch, taking help from torchvision and lightning modules for the training loop. After doing a grid search on different scales (e.g. 10 to 1000 for epoch) for the following hyperparameters (as shown in **Table 1**), we ultimately found best performance to be: batch size of 256, hidden dimension (final embedding size) of 128, learning rate of $5e^{-4}$, temperature of 0.07, weight decay of $1e^{-4}$, and 500 epochs, though the best model checkpoint was achieved at epoch 491 in the ReLU model and epoch 496 in the softmax model. Both models converged, and though accuracy is not relevant to the final clustering goal, loss and accuracy plots from training and validation are provided in **Figure 3** (listed in section **4.4**).

4.3 Clustering

Based on literature, we decided to cluster the audio embeddings for the unseen test data using spectral clustering with cluster-qr for k=number of labels (number of distinct speakers). We also tried clustering on the additional test dataset (JL corpus) with k = 4, but results are not provided due to limited space in the paper. Though there are more refined and nuanced clustering methods using SOTA work, spectral clustering is known to still be reasonably effective for speaker diarization, and since the primary focus of our project is generating embeddings, we determined this to be sufficient.

4.4 Experiments, Evaluations, and Metrics

We then computed the diarization error rate (DER) on the clustered embeddings, which is the most widely used diarization metric[7]. DER is the ratio between the sum of false alarm, missed detection and confusion, to total duration of speech:

$$\text{DER} = \frac{\text{confusion + missed + false}}{\text{total duration of speech}}$$

Even though our test data contains single speaker audios, we simulated a continuous stream of audio by joining together single-speaker audios with random segments of silence to model the speaker diarization use case where multiple speakers are present in a single audio.

We had two sets of hyperparameters to tune – one for data augmentations, and one for model training. The former was set using values that were successful in other similar works. For the latter set of hyperparameters, we performed a grid search across different values to reach an optimal setting. The list of hyperparameters, their meaning, and final values are outlined in **Table 1**. Accuracy and loss plots for for both self-supervised contrastive models (orange - ReLU, blue - softmax) in 500 epochs are shown in **Figure 3**.

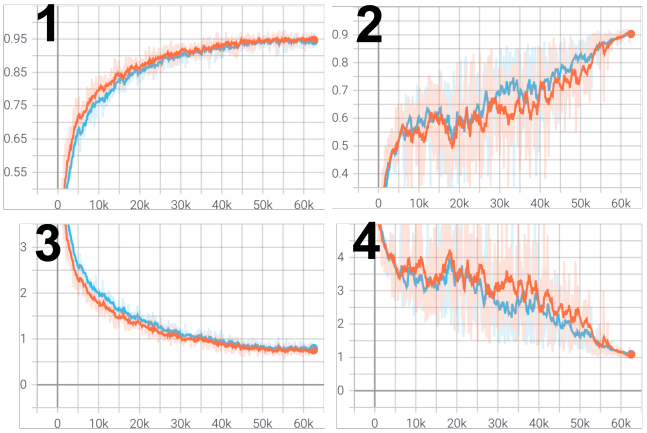

Figure 3: 1: Train Accuracy, 2: Validation Accuracy, 3: Train Loss, 4: Validation Loss

5 Results

Diarization Error Rates on Unseen Test Data	
Embedding Technique	DER
LSTM d-vector (ReLU)	.667
LSTM d-vector (Softmax)	.455
Contrastive (ReLU)	.205
Contrastive (Softmax)	**.189**

Table 2: Diarization Error Rates

From the DERs computed on the test set, we can see that the Softmax self-supervised model achieves the lowest DER. Overall, the self-supervised method does better than the baselines, and the softmax versions do better than the ReLU versions. Though we are not exactly sure why this is the case, training on different hyperparameter permutations for these activations support the conclusion that softmax versions outperform ReLU versions with DER as our metric. SOTA DERs have been as low as 0.12, but they use more complex clustering methods [5]. Since we are using a simple spectral clustering but achieve a DER of 0.189, this indicates that the self-supervised architecture we have devised holds promise if used in tandem with a better clustering method.

6 Future Work

Overall, this approach can be further bettered by incorporating diffusion or momentum distillation, experimenting with MOCO-adjacent architectures, and further tuning the augmentation and training hyperparameters. Our datasets also didn't have much noise, which we corrected for by adding some random noise in the augmentations, but fine-tuning this

model with data that has background noise would also make it more useful for real life deployment. One potential pitfall of this approach is that it primarily explores a vision-based self-supervised approach, but perhaps a better model might use a transformer before the MLP and accept augmented views of an MFCC as the input instead of spectrograms. Though we are not entirely certain what those augmentations would be, it is worth experimenting with.

7 Contributions

Both team members contributed equally in writing the preprocessing and model code (pair coding) as well as the writeup. We would also like to thank our TA, Skanda Vaidyanath, for all the helpful feedback and support!

References

[1] Team Symbl. "What Is Speaker Diarization?" Symbl.ai, 31 Aug. 2022, symbl.ai/blog/what-is-speaker-diarization/.

[2] Aronowitz, Hagai, and Weizhong Zhu. "New Advances in Speaker Diarization." IBM Research Blog, 5 Nov. 2020, www.ibm.com/blogs/research/2020/10/new-advances-in-speaker-diarization/.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Kamarudin, Noraziahtulhidayu, et al. "Feature Extraction Using Spectral Centroid and Mel Frequency Cepstral Coefficient for Quranic Accent Automatic Identification." 2014 IEEE Student Conference on Research and Development, 2014, doi:10.1109/scored.2014.7072945.

[4] Doddipatla, Rama, et al. "Speaker Adaptation in DNN-Based Speech Synthesis Using D-Vectors." Interspeech 2017, 2017, doi:10.21437/interspeech.2017-1038.

[5] Dissen, Y., Kreuk, F., Keshet, J. (2022, April 8). Self-supervised speaker diarization. arXiv.org. Retrieved December 9, 2022, from https://arxiv.org/abs/2204.04166

[6] Google. (n.d.). Detect different speakers in an audio recording | cloud speech-to-text documentation | google cloud. Google. Retrieved December 9, 2022, from https://cloud.google.com/speech-to-text/docs/multiple-voices

[7] C Sailaja et al 2021 J. Phys.: Conf. Ser. 1804 012166

[8] A review of speaker diarization: Recent advances with Deep Learning - arXiv. (n.d.). Retrieved December 10, 2022, from https://arxiv.org/pdf/2101.09624

[9] Learning embeddings for speaker clustering based on Voice Equality. (n.d.). Retrieved December 10, 2022, from https://www.researchgate.net/publication/321785520_Learning_embeddings_for_speaker_clustering_based_on_voice_equality

[10] Speaker diarization. Speaker Diarization - NVIDIA NeMo. (n.d.). Retrieved December 9, 2022, from https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/asr/speaker$_diarization/intro.html$

[11] Speaker diarization using Deep Neural Network embeddings - danielpovey.com. (n.d.). Retrieved December 10, 2022, from https://www.danielpovey.com/files/2017$_icassp_diarization_embeddings.pdf$