Transactions of the International Society for
Music Information Retrieval

Andersen, A. Ryan, A. (2025). Implementing Query by Voice Imitation using PyTorch, *Transactions of the International Society for Music Information Retrieval*, V(N), pp. xx–xx, DOI: https://doi.org/xx.xxxx/xxxx.xx

ARTICLE TYPE

Implementing Query by Voice Imitation using PyTorch

Anthony Ryan Alex Andersen

Abstract

Retrieving sound effects using text-based search methods is often inefficient and imprecise, as textual descriptions may not fully capture the nuances of a sound. Query-by-Vocal-Imitation (QVI) offers an intuitive alternative by allowing users to vocally imitate a sound to retrieve similar audio samples.

Keywords: Query-by-Voice-Imitation, Siamese Neural Network, PyTorch, Sound Retrieval

1. Introduction

Searching for sound effects is an essential task in music production, video editing, game development and more, but this search traditionally relies on text-based tags and keywords, which can be inefficient and subjective. Describing a specific sound—such as a mechanical whirr, an animal call, or an abstract noise-using words is inherently ambiguous, leading to imprecise or time-consuming search results.

Query-By-Vocal-Imitation (QVI) is an exciting application of MIR, and a great alternative to searching using text. Retrieving sounds by mimicking them vocally is faster and more expressive. This project aims to implement QVI in Python using PyTorch, using a Siamese Neural Network trained on the Voice Imitation Dataset. We will use Two-Dimensional Fourier Transforms to increase robustness, allowing voices imitating a particular sample at a different pitch to still match correctly.

2. Related Work

"IMISOUND" (Duan et al., 2016), utilized auto-encoders to learn compact representations of sound features, reducing reliance on manually labeled data. Kim et al. (2023) improved QVI performance by applying contrastive learning, aligning vocal imitation embeddings with their corresponding sound representations. Greif et al. (2024) further improved by using an efficient CNN pre-trained on a large general-purpose audio dataset as an embedding model. The embedding space created by this process is then used to determine the contrasts in similarities between reference and imitation sounds.

3. Methodology

The development of this Query-by-Vocal-Imitation (QVI) system will combine traditional signal processing techniques with Siamese-style neural networks to match vocal imitations to reference sounds. Initially, audio features are extracted using a 2D Fourier transform and constant-Q spectrograms. The feature extraction process begins with applying the Constant-Q Transform (CQT) using torchaudio's built-in transform. CQT preserves the spacing between overtones when the fundamental frequency changes, making it particularly suitable for vocal imitations. Following the CQT, we'll apply a 2D Fourier Transform (2DFT) using `torch.fft.fft2` to capture the spectro-temporal modulation patterns in the audio. A crucial property of the 2DFT is that its magnitude is invariant to frequency and time shifts, helping the model to match the fundamental pattern of the sound regardless of exactly where it starts or what base pitch it's centered around.

Once features are extracted, they will be input into a Siamese neural network, which consists of two identical branches that process the vocal imitation and reference sound. The similarity between the inputs can then be computed using a distance metric such as cosine similarity, and are mapped into a shared embedding space. By using a contrastive loss function such as NT-Xent, we can minimize the distance between embeddings of matching imitation-reference pairs while maximizing the distance between non-matching pairs. The training will also leverage data augmentation to simulate real-world variations in pitch, rhythm, and amplitude, improving the system's robustness.

The user interface, implemented using `tkinter`, will provide a straightforward way to record vocal imitations and view results. This includes a display of potential matches ranked by similarity, and category selection for focused searching. The interface should also incorporate a feedback mechanism to help evaluate the system's performance and potentially collect data for future model improvements.

4. Project Timeline

February 27th - Initial Model Development
The project will begin with setting up the development environment and constructing the model architecture (Ryan). Feature extraction methods will be implemented (Andersen), followed by the first training iteration to establish baseline performance.

March 18th - Project Update
An analysis of initial training results will be conducted, with documentation of the technical approach, challenges, and planned improvements (Andersen). Visualizations of current performance will be created, and the methodology section will be drafted (Ryan).

March 22th - Real-time Audio Processing (Basic)
Focus will shift to audio input handling and preprocessing (Andersen), alongside building the storage system (Ryan). Feature extraction will be enhanced, and data augmentation techniques implemented (Ryan). Evaluation metrics will be defined, and an automated testing pipeline will be developed (Andersen). This is the baseline for what we can expect to get done.

March 25th - Model Optimization (Expected)
This phase will involve hyperparameter tuning, testing similarity metrics, and optimizing model performance to improve accuracy and efficiency (Andersen, Ryan). Documentation and a demo version will be prepared (Andersen). This is what we expect to get done.

March 29th - System Integration & Testing (Advanced)
System integration will be carried out, with testing for compatibility and performance. Any issues will be addressed, and optimization will continue (Ryan). This is what we hope to achieve if all goes well.

April 2nd - Interface Development (Advanced)
The user interface will be developed, including an audio recording interface and visualization of matched results. A feedback mechanism and basic error handling will be integrated (Andersen). This is what we hope to achieve if all goes well.

April 3rd - Project Paper
The final phase will include analysis of results and performance metrics, documentation of the methodology, and a literature review. The project report, including future work, an abstract, and an introduction, will be completed (Andersen, Ryan).

5. Process

Our system uses a contrastive learning approach to create a shared embedding space where vocal imitations and their corresponding reference sounds are positioned in proximity to one another Greif et al. (2024). This allows the model to retrieve matching sounds given a vocal imitation query.

We structured our development into four main steps: Data Preparation, Model Setup, Training Loop, and Evaluation. Thus far, we have implemented the data preparation stage and feature extraction for pre-training, following the methodology of Greif et al. (2024).

5.1 Data Structure

Our dataset preserves the relationships between reference sounds and their vocal imitations. Each entry consists of a reference sound file path and a list of corresponding imitation file paths. We parsed the dataset into a list of tuples where the first element is the reference file and the second is the list of its imitation files. This list was then split into two subsets: training and validation. This split ensures that reference sounds and their associated imitations in the validation set are unseen during training, allowing us to evaluate generalization performance. We also defined batch sizes and shuffle settings for the data loaders to facilitate efficient and randomized access during training.

5.2 Feature Extraction

Our feature extraction pipeline is designed to match the preprocessing used by the pre-trained MobileNetV3 model, following Greif et al. (2024). Specifically, we convert audio files into mel-spectrograms using the same parameters as those used during MobileNetV3's original training on AudioSet.

The resulting mel-spectrograms are represented as tensors with shape (B, C, T, F), where B is the batch size, C is the number of channels, T is the number of time frames, and F is the number of mel frequency bins. This shape aligns with the input expected by the MobileNetV3 architecture.

5.3 Data Augmentation

Data augmentation is commonly used to improve model robustness by introducing variations in the training data. For our task, augmentation is especially useful because vocal imitations can vary widely in tone, rhythm, and noise levels. By augmenting both reference sounds and imitations, we can help the model generalize across different imitation styles and recording conditions.

Following Greif et al. (2024), we implemented several augmentation techniques, including time masking, frequency masking, and time stretching. These techniques alter the mel-spectrograms in controlled ways, effectively increasing the diversity of training examples without requiring additional data collection. Example augmentations are illustrated in Figure 1.

5.4 Viewing Data

To verify the integrity of our pipeline, we implemented tools to view statistics on the data batches, generate spectrogram visualizations, and reconstruct audio from spectrograms. These diagnostics allowed us to confirm that feature extraction and augmentation were functioning as intended. For example, we visually inspected spectrograms to ensure they had the expected structure and confirmed that reconstructed audio re-

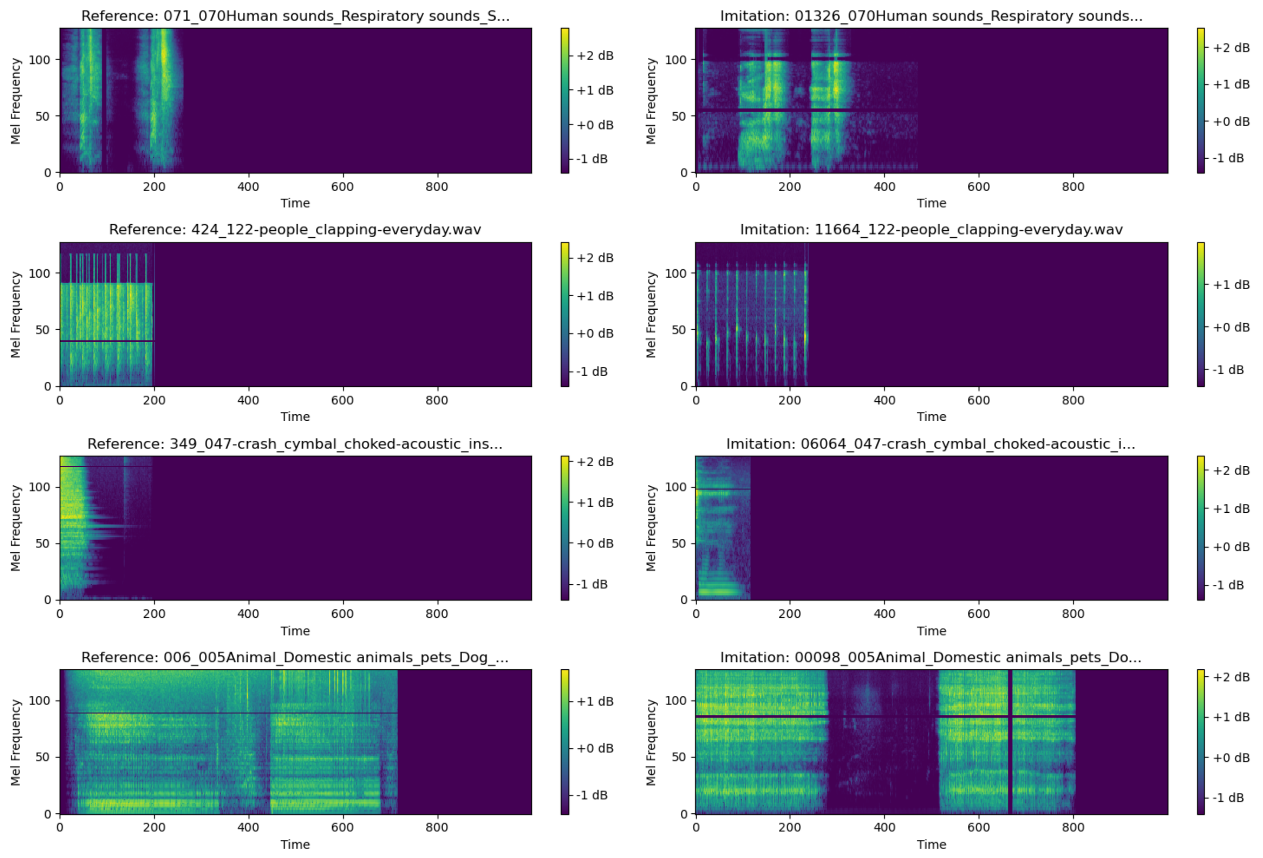

Figure 1: Spectrograms of four randomly selected reference-imitation pairs after data augmentation from batch

tained key perceptual features. This inspection process helped identify and correct any preprocessing mismatches early in development.

6. Planned Work: Training Loop and Evaluation

Our next step is to implement the training loop using a Siamese network architecture with shared MobileNetV3 encoders for reference and imitation inputs. We will fine-tune the network using contrastive loss (NT-Xent) to bring matching reference–imitation pairs closer in the embedding space, while pushing non-matching pairs apart.

During each training iteration, we will construct batches that include multiple reference–imitation pairs. For each pair, one imitation serves as the positive example, and other non-matching references and imitations serve as negatives. This structure allows the contrastive loss to learn meaningful distinctions.

In the evaluation phase, we will assess the model's ability to retrieve reference sounds given a vocal imitation. Metrics such as Top-N retrieval accuracy and mean reciprocal rank will quantify performance. We also plan to conduct qualitative evaluations by listening to retrieved sounds to test for perceptual relevance. This evaluation will be performed on a held-out test set for fair assessment of generalization capability.

References

Bansal, A. and Garg, N. K. (2022). Environmental sound classification: A descriptive review of the literature. *Intelligent Systems with Applications*, 16:200115.

Bongjun Kim, Madhav Ghei, B. P. Z. D. (2018). Vocal imitation set: A dataset of vocally imitated sound events using this audioset ontology. In *Detection and Classification of Acoustic Scenes and Events 2018*.

Greif, J., Schmid, F., Primus, P., and Widmer, G. (2024). Improving query-by-vocal imitation with contrastive learning and audio pretraining. In *Detection and Classification of Acoustic Scenes and Events (DCASE)*, New York, NY, USA.

Grosche, P., Müller, M., and Serrà, J. (2012). Audio Content-Based Music Retrieval. In Müller, M., Goto, M., and Schedl, M., editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 157–174. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany.

Jonathan Greif, Florian Schmid, P. P. G. W. (2024). Improving query-by-vocal imitation with contrastive learning and audio pretraining. *arXiv*.

Kim, B. and Pardo, B. (2019). Improving content-based audio retrieval by vocal imitation feedback. In *Proceedings of the IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK. IEEE.

Kurth, F. and Muller, M. (2008). Efficient index-based audio matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):382–395.

Peng, Y., Ngo, C.-W., Fang, C., Chen, X., and Xiao, J. (2006). Audio similarity measure by graph modeling and matching. In *Proceedings of the 14th ACM International Conference on Multimedia*, MM '06, page 603–606, New York, NY, USA. Association for Computing Machinery.

Pishdadian, F., Kim, B., Seetharaman, P., and Pardo, B. (2019). Classifying non-speech vocals: Deep vs signal processing representations. In *Detection and Classification of Acoustic Scenes and Events (DCASE)*, New York, NY, USA.

Song, J., Bae, S.-Y., and Yoon, K. (2002). Query by humming: matching humming query to polyphonic audio. In *Proceedings. IEEE International Conference on Multimedia and Expo*, volume 1, pages 329–332 vol.1.

Yuki Okamoto, Keisuke Imoto, S. T. R. N. T. F. Y. Y. (2023). Environmental sound synthesis from vocal imitations and sound event labels. *arXiv*.

Zhang, T. and Kuo, C.-C. (1999). Classification and retrieval of sound effects in audiovisual data management. In *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers (Cat. No.CH37020)*, volume 1, pages 730–734 vol.1.

Zhang, Y. and Duan, Z. (2015). Retrieving sounds by vocal imitation recognition. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6.

Zhang, Y. and Duan, Z. (2017). Iminet: Convolutional semi-siamese networks for sound search by vocal imitation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 304–308.

Zhang, Y., Hu, J., Zhang, Y., Pardo, B., and Duan, Z. (2020). Vroom! a search engine for sounds by vocal imitation queries. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, CHIIR '20, page 23–32, New York, NY, USA. Association for Computing Machinery.