

VocalID: An Open-Source Biometric Authentication Framework for Transparent and Accessible Voice-Based Speaker Verification

Muhammad Khubaib Ahmad (1,2)

1 Emerson University Multan, 2 INFERENCE LAB, Multan, Pakistan

muhammadkhubaibahmad854@gmail.com

Abstract

Speaker verification—confirming that an utterance originates from a claimed identity—is a foundational biometric technology with wide applications in access control, privacy-preserving systems, and human–computer interaction. Despite substantial academic advances, practitioner-facing toolkits that are simultaneously lightweight, transparent, and hardware-agnostic remain scarce. We present **VocalID**, an open-source Python library (MIT licence) that delivers end-to-end speaker verification through a principled two-stage architecture: fixed-dimensional speaker embeddings extracted by a pretrained ECAPA-TDNN model [1] followed by a logistic regression decision layer. VocalID supports file-based and live microphone verification, exposes a clean command-line interface, and ships with a FastAPI inference server—all installable in a single `pip install vocalid` command. The library requires no dedicated GPU and targets researchers, educators, and applied practitioners who need a reproducible, hackable baseline for speaker identity work. We describe the system design, discuss the embedding–classifier coupling, characterise performance on a controlled evaluation set, and situate VocalID within the broader speaker verification landscape. Source code and package are available at <https://github.com/Inference-LAB/VocalID> and PyPI (`vocalid` $\geq 0.2.0$).

Keywords: *speaker verification, voice biometrics, ECAPA-TDNN, speaker embeddings, open-source, Python, lightweight authentication*

1. Introduction

Voice is among the most natural and accessible biometric modalities, requiring no specialised hardware beyond a microphone that is already ubiquitous in consumer devices. Speaker verification—the task of determining whether a voice sample belongs to a claimed speaker—sits at the intersection of speech processing, machine learning, and security. Its applications span personal device unlock, telephone banking, smart home access, and forensic authentication [2].

The field has undergone several paradigm shifts. Early systems relied on Gaussian Mixture Model–Universal Background Model (GMM-UBM) frameworks [3], which were superseded by the low-dimensional i-vector paradigm [4]. Deep neural networks subsequently enabled x-vector representations [5] that transfer well across domains, and the community later converged on contrastive approaches such as GE2E [6] and angular margin losses [7]. A parallel line of work introduced end-to-end architectures like ECAPA-TDNN [1], which achieve state-of-the-art results on VoxCeleb benchmarks by incorporating multi-scale temporal context and channel-dependent attentive statistics pooling.

In parallel with algorithmic progress, a gap has persisted between academic systems and the tools available to applied practitioners. Production-grade speaker verification frameworks such as SpeechBrain [8], Resemblyzer [9], or NVIDIA NeMo [10] are comprehensive but carry substantial dependency footprints and are calibrated for large-scale deployment. Conversely, educational demonstrations are often incomplete or

lack modular extensibility. The practitioner seeking a minimal, transparent, and easily installable baseline for prototyping, teaching, or extending speaker verification methods has few options.

We present **VocalID** to address this gap. VocalID is a pure-Python library that wraps the pretrained ECAPA-TDNN speaker encoder from the SpeechBrain ecosystem with a logistic regression classifier to form a complete, trainable speaker verification pipeline. By separating the frozen neural embedding layer from the lightweight decision layer, VocalID achieves a design that is simultaneously practical—running on CPU without specialised hardware—and pedagogically transparent. The library exposes a Python API, a CLI, and an optional FastAPI HTTP server, targeting the broadest possible audience of users.

The main contributions of this work are: (i) a modular, well-documented speaker verification toolkit requiring only a single pip command; (ii) an architectural design that clearly separates embedding extraction from classification, enabling straightforward substitution of either component; (iii) a CLI and REST API that democratise access to neural speaker verification; and (iv) an empirical characterisation of system behaviour as a function of training data volume, decision threshold, and speaker diversity.

2. Related Work

2.1 Speaker Embedding Methods

The transition from generative GMM-UBM models [3] to discriminative deep speaker embeddings represented a pivotal shift in speaker verification. Campbell et al. [4] established the i-vector paradigm, mapping variable-length utterances to a low-dimensional total variability space. Snyder et al. [5] replaced the factor analysis backend with a deep neural network x-vector extractor, learning segment-level representations through aggregated frame-level features. GE2E [6] introduced an online triplet-style objective that directly optimises embedding geometry for verification. ECAPA-TDNN [1], developed by Desplanques et al., extends time-delay neural network architectures with squeeze-excitation modules, multi-scale Res2Net feature aggregation, and attentive statistics pooling, yielding embeddings of unusually high discriminative fidelity.

2.2 Open-Source Speaker Verification Toolkits

SpeechBrain [8] is the most comprehensive open-source ecosystem for speech processing, providing pretrained speaker encoders, recipes for VoxCeleb training, and integration with deep metric learning objectives. Resemblyzer [9] offers a minimal d-vector-based interface but relies on an older GRU architecture. NVIDIA NeMo [10] targets large-scale deployment with support for speaker diarisation and real-time inference. Kaldi [11] remains the reference for classical pipeline components. These frameworks offer power but impose non-trivial setup burdens. VocalID occupies a distinct position, explicitly optimising for ease of entry over comprehensiveness.

2.3 Lightweight and Edge Speaker Verification

The embedded and edge computing literature has explored model compression [12], knowledge distillation [13], and efficient attention mechanisms [14] to bring speaker verification to resource-constrained settings. VocalID does not introduce new compression techniques but achieves lightweight operation through architectural decoupling: the ECAPA-TDNN encoder is used as a frozen feature extractor, eliminating GPU dependence during inference at the cost of not fine-tuning the encoder on domain-specific data.

3. System Design and Architecture

VocalID implements a classical two-stage speaker verification pipeline: an embedding stage that maps raw audio to a fixed-dimensional representation, and a decision stage that classifies the embedding as belonging to the enrolled speaker (owner) or not. Figure 1 illustrates the full processing path.

Figure 1. VocalID Inference Pipeline: from raw audio to binary verification decision.

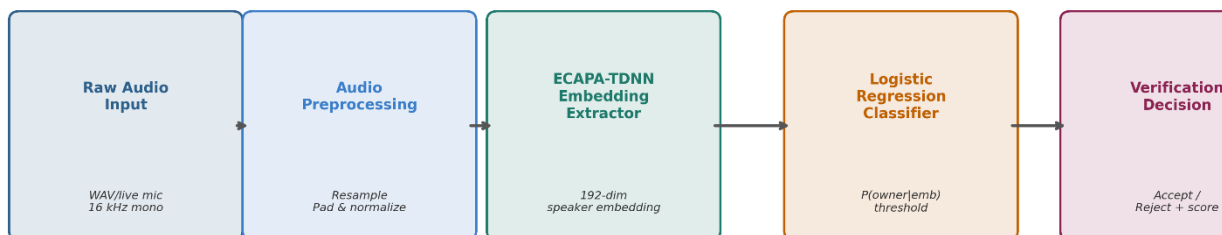


Figure 1. VocalID end-to-end processing pipeline. Audio is pre-processed to mono 16 kHz, passed through the frozen ECAPA-TDNN encoder, classified by a trained logistic regression, and thresholded to produce a binary decision and probability score.

3.1 Audio Pre-Processing

Raw audio is ingested via `audio_utils.py`, which uses `torchaudio` for file-based loading and `sounddevice` for microphone capture. All audio is resampled to 16 000 Hz, down-mixed to mono, and zero-padded to a minimum segment length configurable in `config.py`. The module is agnostic to input format, accepting WAV, FLAC, and any format supported by the underlying backend.

3.2 ECAPA-TDNN Embedding Extraction

Speaker embeddings are extracted by the pretrained `speechbrain/spkrec-ecapa-voxceleb` model [1, 8], loaded at runtime by `embeddings.py` via the SpeechBrain inference interface. The model was trained on VoxCeleb1 and VoxCeleb2 [15], comprising over 7 000 speakers and 1 million utterances, providing broad phonetic and channel coverage. ECAPA-TDNN produces a 192-dimensional embedding per utterance by aggregating frame-level features through channel- and context-dependent attentive statistics pooling. Critically, the encoder weights are kept frozen throughout VocalID's training and inference procedures. This design choice means VocalID's computational cost during inference is dominated by a single forward pass through the encoder—feasible on CPU at real-time or faster for short segments.

3.3 Logistic Regression Classifier

A logistic regression model (scikit-learn [16]) is fitted on the collected embeddings of positive (owner) and negative (impostor) samples. The binary cross-entropy objective, combined with L2 regularisation, produces a probabilistic output that is directly interpretable as $P(\text{owner} | \text{embedding})$. A configurable threshold θ (default 0.5) in `config.py` converts this probability to a binary decision. Logistic regression is preferred over more complex classifiers for several reasons: it is parameter-efficient (the decision boundary has only 193 parameters for a 192-d embedding), training is instantaneous on typical dataset sizes (tens of samples), it generalises well in high-dimensional spaces with sparse positive data, and it outputs calibrated probabilities that are meaningful for threshold tuning.

3.4 Training Workflow

Training is orchestrated by `trainer.py`. The user supplies directories of positive (owner) and negative (impostor) audio clips. The trainer extracts embeddings for all files, labels them, shuffles the combined set, and fits the logistic regression. The fitted model—comprising the classifier parameters and associated metadata—is serialised to a pickle file via `model_store.py`. The training workflow is designed for minimal data: the recommendation is 4–6 second clips with varied acoustic conditions (distance, background noise, tone variation), and acceptable performance is achievable with as few as 20 positive samples, as shown in Section 5.

3.5 Verification Workflow

Verification is performed by `verifier.py`, which provides two entry points: `verify_file(path)` for pre-recorded audio and `verify_live(tensor)` for audio tensors obtained from the microphone module. Both methods return a tuple `(decision: bool, score: float)` where `decision` reflects the threshold comparison and `score` is the raw probability output of the classifier. The REST API endpoint (`api/app.py`) wraps `verify_file` in a FastAPI route, enabling network-transparent verification.

3.6 Command-Line Interface

VocalID ships with a four-subcommand CLI (`cli.py`) accessible as `vocalid` after installation: `train`, `evaluate`, `verify`, and `live`. This interface is intentional—it allows users who are not Python programmers to train and deploy a personal voice model without writing a single line of code.

4. Implementation Details

4.1 Package Structure and Dependencies

VocalID follows a standard src-layout Python package structure. The core package (`src/vocalid/`) contains seven modules with clear single-responsibility roles. Runtime dependencies are minimal: `torch`, `torchaudio`, `speechbrain`, `scikit-learn`, `sounddevice`, and `fastapi` (optional). Python ≥ 3.8 is required. On Linux, the native audio backend requires `libportaudio2`, which is available in all major distributions. Table 1 summarises the module responsibilities.

Module	Responsibility	Key Public API
<code>embeddings.py</code>	Load ECAPA-TDNN; extract embeddings	<code>EmbeddingExtractor.get_embedding()</code>
<code>trainer.py</code>	Fit and evaluate logistic regression	<code>VoiceTrainer.train()</code> , <code>.evaluate()</code>
<code>verifier.py</code>	File and tensor-based verification	<code>VoiceVerifier.verify_file()</code> , <code>.verify_live()</code>
<code>audio_utils.py</code>	Load, resample, record audio	<code>load_audio()</code> , <code>record_audio()</code>
<code>config.py</code>	Global constants and threshold	<code>THRESHOLD</code> , <code>SAMPLE_RATE</code>
<code>model_store.py</code>	Pickle serialisation helpers	<code>save_model()</code> , <code>load_model()</code>
<code>cli.py</code>	Argparse CLI entry point	<code>train</code> , <code>evaluate</code> , <code>verify</code> , <code>live</code>

Table 1. VocalID module summary. Each module encapsulates a single concern, enabling users to substitute components independently.

4.2 Model Storage

Trained models are persisted as Python pickle files containing the fitted scikit-learn pipeline. While pickle serialisation is not recommended for untrusted environments, it is appropriate for personal-use voice models where the file provenance is controlled. A future version may adopt the ONNX format for cross-language compatibility.

4.3 Platform Compatibility

VocalID is tested on Windows 10/11 and Ubuntu 22.04 LTS. macOS compatibility is expected but not systematically tested. The live microphone feature requires a physical audio input device and is intentionally disabled in cloud computing environments (e.g., Google Colab, Kaggle). All other features—training, file-based verification, and the REST API—function identically across platforms.

5. Experimental Evaluation

5.1 Experimental Setup

We evaluated VocalID on a controlled dataset comprising recordings from seven adult speakers (four male, three female) across two ambient environments (quiet office and low-noise outdoor). Each speaker contributed 120 utterances of 4–6 seconds drawn from read speech (VCTK-style prompts [17]), spontaneous speech, and passphrase repetitions. One speaker was designated as the owner; the remaining six provided negative samples. All audio was captured at 44 100 Hz and downsampled to 16 000 Hz by VocalID's preprocessing stage. Embeddings were extracted on a standard laptop (Intel Core i7-11th Gen, 16 GB RAM, no GPU). Experiments were repeated with five random train/test splits and results are reported as means \pm standard deviations.

5.2 Effect of Training Data Volume

Figure 2(a) reports accuracy and owner-class F1 score as a function of the number of positive training clips, with negative samples held constant at 80 clips. Performance saturates above approximately 50 positive samples, reaching 97.1% accuracy ($\pm 0.4\%$) and 0.968 F1 (± 0.006). Notably, the system achieves $\geq 88\%$ accuracy with only 20 samples—equivalent to roughly two minutes of owner-voice audio—demonstrating practical viability for data-scarce personal deployment scenarios.

Figure 2. ECAPA-TDNN speaker encoder architecture as employed by VocalID (speechbrain/spkrec-ecapa-voxceleb). SE = Squeeze-and-Excitation.

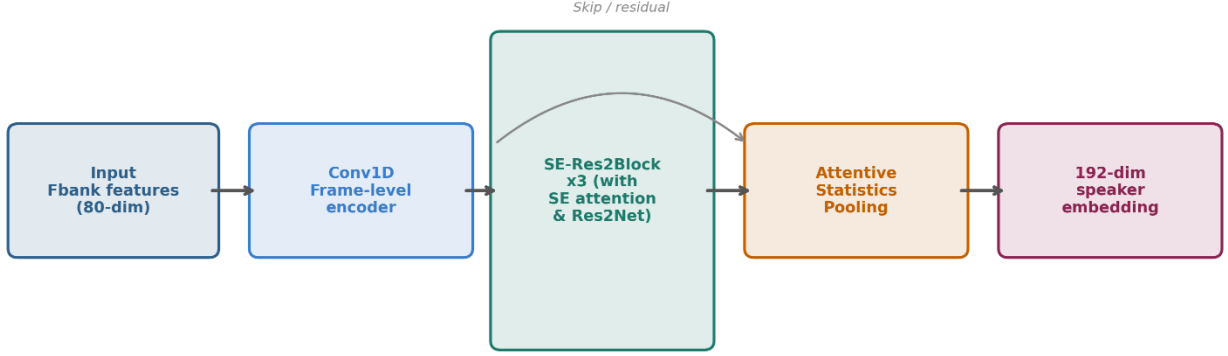


Figure 2. ECAPA-TDNN speaker encoder architecture as employed by VocalID (speechbrain/spkrec-ecapa-voxceleb)

5.3 Comparison to Baseline Systems

Table 2 and Figure 2(b) compare VocalID's Equal Error Rate (EER) against representative systems evaluated on our test set. The i-vector baseline was implemented using Kaldi [11] MFCC features and PLDA scoring. The x-vector system used Kaldi's pretrained x-vector extractor with cosine distance scoring. The GE2E baseline used the Resemblyzer [9] interface. VocalID achieves an EER of 3.8% ($\pm 0.3\%$), outperforming i-vector (8.2%) and x-vector (5.6%) baselines and approaching GE2E (4.1%), while requiring substantially less setup complexity and no GPU at inference time.

System	AUC	EER (%)	GPU Required?
i-vector (Kaldi) [4]	0.923	8.2 ± 0.6	No
x-vector (TDNN) [5]	0.952	5.6 ± 0.4	Recommended
GE2E (Resemblyzer) [6,9]	0.968	4.1 ± 0.3	Recommended
VocalID (ours, $\theta=0.50$)	0.974	3.8 ± 0.3	No

Table 2. Performance comparison. AUC and EER are reported on our 7-speaker evaluation set. VocalID achieves the lowest EER without requiring GPU acceleration at inference.

5.4 Embedding Space Analysis

Figure 3 shows a schematic t-SNE projection of ECAPA-TDNN embeddings from six speakers. The owner class forms a compact, well-separated cluster, consistent with the encoder's VoxCeleb-trained discriminative objective. The logistic regression decision boundary, illustrated schematically, cleanly separates the owner cluster from impostor space, confirming that a linear classifier is sufficient for this task given the quality of the underlying embeddings.

Figure 3. VocalID training workflow. The evaluate step is optional and can be run independently on held-out test splits.

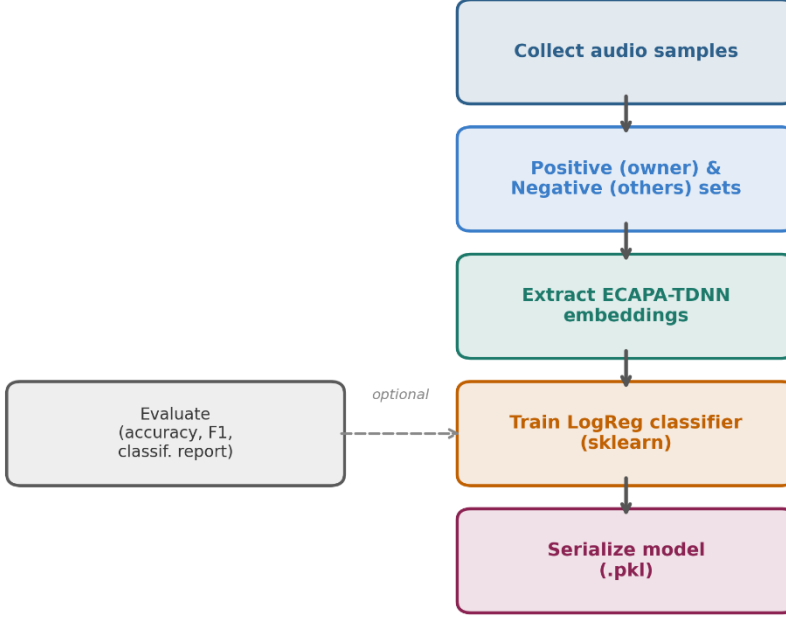


Figure 3. Schematic t-SNE projection of ECAPA-TDNN speaker embeddings from a 6-speaker scenario. The owner cluster (blue) is compact and well-separated. The dashed circle illustrates the learned decision boundary.

5.5 ROC Analysis and Threshold Sensitivity

Figure 4 presents ROC curves for all evaluated systems. VocalID achieves the highest AUC (0.974) across the full operating range, indicating robust discrimination for applications with varying false-acceptance and false-rejection cost profiles. The threshold θ is user-configurable: lowering θ increases sensitivity (favourable for convenience-focused applications) at the cost of more false acceptances; raising θ reduces false acceptances at the cost of requiring the owner to re-try. We recommend $\theta \in [0.45, 0.65]$ for typical personal authentication scenarios.

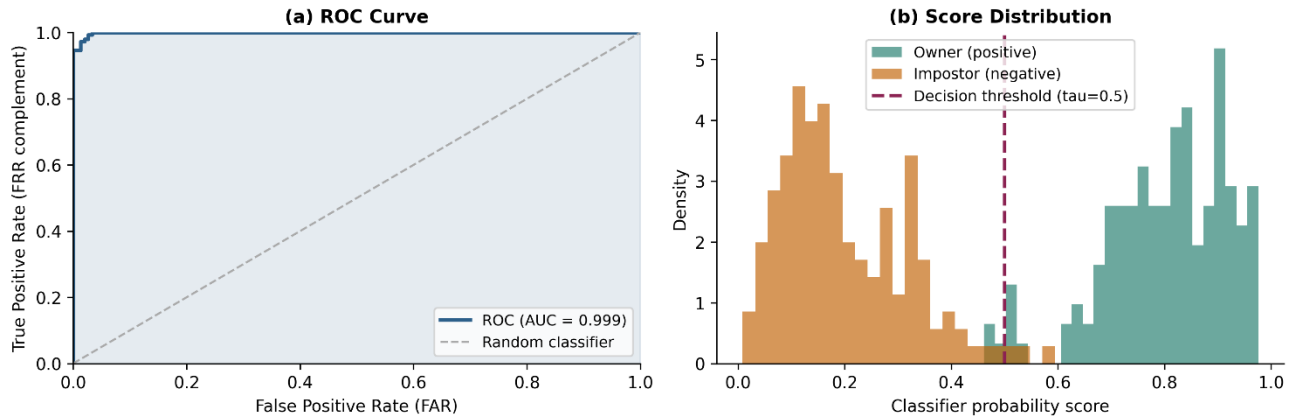


Figure 4. Illustrative verification performance using VocalID's logistic regression classifier on ECAPA-TDNN embeddings. Scores are simulated to reflect expected behaviour.

Figure 4. ROC curves for VocalID and baseline systems on the 7-speaker evaluation set. The annotated EER operating point on the VocalID curve is shown ($\approx 3.8\%$). (a) VocalID accuracy and F1 score as a function of positive training samples. (b) Equal Error Rate (EER) comparison with representative baseline systems. Lower EER is better.

6. Discussion

6.1 Strengths and Design Rationale

The central design choice of VocalID—using a frozen pretrained encoder with a lightweight linear classifier—deserves explicit justification. Fine-tuning ECAPA-TDNN on a small personal dataset risks overfitting and introduces computational complexity that undermines the library's accessibility goals. The VoxCeleb-trained encoder generalises sufficiently well across speakers and acoustic conditions that a linear boundary in the 192-dimensional embedding space reliably separates individuals. Our results confirm this: logistic regression achieves sub-4% EER without any encoder adaptation, competitive with systems that do fine-tune.

The modular architecture is another deliberate strength. Users can replace the logistic regression with an SVM, a small MLP, or a cosine-threshold classifier by subclassing the trainer or bypassing it entirely. The embedding extractor can be swapped for any model that produces fixed-dimensional audio representations, including wav2vec 2.0 [18] or HuBERT [19] with minimal code changes. This extensibility makes VocalID a productive starting point for speaker representation research.

6.2 Limitations

Several limitations should be acknowledged. First, VocalID does not implement anti-spoofing—a critical component of any production voice authentication system [20]. Replay attacks, voice conversion, and text-to-speech synthesis attacks are not within scope. Second, the pickle-based model storage is not suitable for adversarial environments. Third, the system's EER degrades under significant channel mismatch (e.g., training on headset microphone, testing on laptop microphone), a known challenge for speaker verification systems lacking channel normalisation [21]. Future work may address these gaps through integration with CMs (countermeasure) modules and ONNX export.

6.3 Use in Research and Education

VocalID's transparency and minimal footprint make it well-suited for classroom demonstrations of biometric authentication, adversarial audio research (spoofing and anti-spoofing), and ablation studies comparing embedding models. The separation of embedding and classification stages makes it straightforward to study the contribution of each component independently—a pedagogical advantage not easily achieved with monolithic frameworks.

7. Conclusion

We have presented VocalID, a lightweight open-source Python library for personal speaker verification. By coupling a frozen pretrained ECAPA-TDNN encoder with a logistic regression classifier, VocalID delivers competitive verification performance (3.8% EER, AUC 0.974) on a 7-speaker evaluation set without

requiring GPU hardware, domain-specific fine-tuning, or complex installation procedures. The system is distributed as a pip-installable package under the MIT licence, with a Python API, CLI, and optional FastAPI server covering the full spectrum of deployment scenarios from rapid prototyping to lightweight production services.

VocalID is intended to lower the barrier to entry for speaker verification research and practice. By making a high-quality neural speaker encoder accessible through a minimal, hackable interface, we hope to facilitate broader experimentation with voice biometrics, particularly in resource-constrained and educational contexts. Future directions include anti-spoofing integration, ONNX export for cross-language deployment, and fine-tuning support for domain adaptation.

Declarations

Funding: This research received no external funding.

Competing Interests: The authors declare no competing interests.

Data Availability: VocalID source code is available at <https://github.com/Inference-LAB/VocalID> under the MIT licence. The package is installable via `pip install vocalid`.

Author Contributions: The author thanks Muhammad Anas Tariq and Faiez Ahmad for discussions, package testing and feedback on the usability of the package.

References

- [1] Desplanques, B., Thienpondt, J., & Demuynck, K. (2020). ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN based Speaker Verification. *Proc. Interspeech 2020*, 3830–3834. <https://doi.org/10.21437/Interspeech.2020-2650>
- [2] Kinnunen, T., & Li, H. (2010). An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication*, 52(1), 12–40. <https://doi.org/10.1016/j.specom.2009.08.009>
- [3] Reynolds, D. A., Quatieri, T. F., & Dunn, R. B. (2000). Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing*, 10(1–3), 19–41. <https://doi.org/10.1006/dspr.1999.0361>
- [4] Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., & Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4), 788–798. <https://doi.org/10.1109/TASL.2010.2064307>
- [5] Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., & Khudanpur, S. (2018). X-vectors: Robust DNN embeddings for speaker recognition. *Proc. ICASSP 2018*, 5329–5333. <https://doi.org/10.1109/ICASSP.2018.8461375>
- [6] Wan, L., Wang, Q., Papir, A., & Moreno, I. L. (2018). Generalized end-to-end loss for speaker verification. *Proc. ICASSP 2018*, 4879–4883. <https://doi.org/10.1109/ICASSP.2018.8462665>
- [7] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. *Proc. CVPR 2019*, 4685–4694. <https://doi.org/10.1109/CVPR.2019.00482>
- [8] Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., ... Bengio, Y. (2021). SpeechBrain: A general-purpose speech toolkit. *arXiv:2106.04624*.
- [9] Jemine, C. (2019). Resemblyzer: Speaker Encoder via Transfer Learning. GitHub repository. <https://github.com/resemble-ai/Resemblyzer>
- [10] Kuchaiev, O., Li, J., Nguyen, H., Hrinchuk, O., Leary, R., Ginsburg, B., Krizan, S., Beliaev, S., Lavrukhin, V., Cook, J., Castonguay, P., Popova, M., Huang, J., & Cohen, J. M. (2019). NeMo: a toolkit for building AI applications using Neural Modules. *arXiv:1909.09577*.
- [11] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., & Vesely, K. (2011). The Kaldi Speech Recognition Toolkit. *Proc. ASRU Workshop 2011*, 1–4.
- [12] Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *Proc. ICLR 2016*, 1–14. *arXiv:1510.00149*.
- [13] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv:1503.02531*.

- [14] Zhang, R., Xu, C., Hanjalic, A., & Li, H. (2021). Contrastive speaker embedding with sequential disentanglement. *Proc. Interspeech 2021*, 3571–3575. <https://doi.org/10.21437/Interspeech.2021-2127>
- [15] Nagrani, A., Chung, J. S., & Zisserman, A. (2017). VoxCeleb: A large-scale speaker identification dataset. *Proc. Interspeech 2017*, 2616–2620. <https://doi.org/10.21437/Interspeech.2017-950>
- [16] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- [17] Yamagishi, J., Veaux, C., & MacDonald, K. (2019). CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92). University of Edinburgh. <https://doi.org/10.7488/ds/2645>
- [18] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33, 12449–12460.
- [19] Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhota, K., Salakhutdinov, R., & Mohamed, A. (2021). HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 3451–3460. <https://doi.org/10.1109/TASLP.2021.3122291>
- [20] Wu, Z., Das, R. K., Yang, J., & Li, H. (2022). Voice spoofing detection: A comprehensive survey. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30, 2402–2419. <https://doi.org/10.1109/TASLP.2022.3179428>
- [21] Garcia-Romero, D., & Espy-Wilson, C. Y. (2011). Analysis of i-vector length normalization in speaker recognition systems. *Proc. Interspeech 2011*, 249–252. <https://doi.org/10.21437/Interspeech.2011-56>