Do Speech Tokens Leak Voiceprints? Speaker Inversion Attacks Against End-to-End Speech Language Models

Ye Lu, Yihan Yan, Zhaoyang Zhang, Zhitao Ou, Runze Liu, Li Liu, and Shen Wang
School of Cyberspace Science, Harbin Institute of Technology, Harbin 150001, China
Corresponding author: Shen Wang (shen.wang@hit.edu.cn)

arXiv:2607.16870v1 [cs.SD] 18 Jul 2026

***Abstract*—End-to-end speech language models increasingly represent user speech with speech tokens rather than relying exclusively on cascaded ASR–LLM–TTS pipelines. Although these tokens support expressive and low-latency spoken interaction, they may also preserve sensitive speaker characteristics. We investigate whether exposed speech tokens leak voiceprints and formulate this risk as a speaker inversion attack. We introduce Audio BERT (AuB), a trainable model that constructs token embeddings from discrete codebooks and aggregates them into speaker-sensitive representations, and propose SpInv, a two-stage inversion method built on AuB to recover embeddings in the space of an attacker-specified speaker encoder. We evaluate Moshi, Higgs3, Kimi-Audio, and Qwen3-Omni using speaker-disjoint protocols on the VoxCeleb dataset. Extensive experiments show that, with only three seconds of frontend output, SpInv achieves cosine similarities above 0.70 in the attacker-specified speaker-encoder space.**

***Index Terms*—End-to-end speech language models, speech tokens, speaker inversion attack, voiceprint privacy, speaker embedding recovery**

I. Introduction

End-to-end speech language models are emerging as a natural interface for interaction with large models. Unlike cascaded ASR–LLM–TTS systems that convert speech into text for language-model processing, these models directly process speech-modality tokens throughout inference. Such representations preserve prosody, emotion, speaking style, and other acoustic cues, enabling emotion-aware and context-sensitive interaction. They also facilitate bidirectional and full-duplex speech modeling while reducing modality-conversion overhead and response latency.

Directly transmitting user speech to cloud-side models can expose voiceprints and other sensitive acoustic attributes. A split-inference deployment may instead run a tokenizer or speech frontend locally and transmit only its output to the cloud-side model, leaving the downstream provider or an intermediate service with speech tokens rather than the waveform. Speech tokens are more compact and task-oriented than raw audio, but it remains unclear whether they suppress speaker identity or merely encode it in a different form. This motivates our central question: *Do speech tokens in end-to-end speech language models leak voiceprints?*

We formulate this privacy question as a Speaker Inversion Attack (SIA) against speech tokens, which extends the concept of a Model Inversion Attack (MIA) [1] to speaker privacy in the speech modality. As illustrated in Fig. 1, the attack arises when an eavesdropper intercepts speech tokens transmitted between the frontend and the downstream model, or when a curious service provider can access them during inference. The adversary then uses the exposed token segment to recover a speaker embedding in the space of a speaker encoder that the adversary specifies. To instantiate this attack, we propose SpInv, a two-stage inversion method that employs Audio BERT (AuB) to aggregate speaker information from speech tokens. AuB maps discrete codebook indices to trainable token embeddings and encodes the resulting sequence with a BERT backbone, while SpInv combines distillation pretraining with discriminative fine-tuning. The trained method recovers a speaker embedding in the attacker-specified speaker-encoder space.

We evaluate four representative speech-model frontends: Moshi, Higgs3, Kimi-Audio, and Qwen3-Omni. The evaluation first tests whether their exposed interfaces retain speaker-discriminative information. It then assesses speaker embedding recovery on speaker-disjoint VoxCeleb datasets followed by analyses of input duration and target speaker space. Our results show that SpInv can recover an embedding in the specified speaker-encoder space from only three seconds of exposed frontend outputs, achieving cosine similarities generally above 0.70 with the target embedding.

The main contributions of this work are as follows:

- To the best of our knowledge, we present the first systematic study of privacy leakage from speech tokens and extend model inversion in the speech modality to the problem of speaker-identity privacy.
- We propose SpInv, a two-stage speaker inversion method that employs an AuB model to construct and aggregate token representations and recover embeddings in the space of an attacker-specified speaker encoder.
- We conduct extensive experiments on four representative target speech frontends, including leakage validation, speaker embedding recovery, and ablations on key factors.

II. Related Work

A. End-to-End Speech Language Models and Speech Interfaces

Early spoken interaction systems built around large language models were commonly modular or text-mediated.

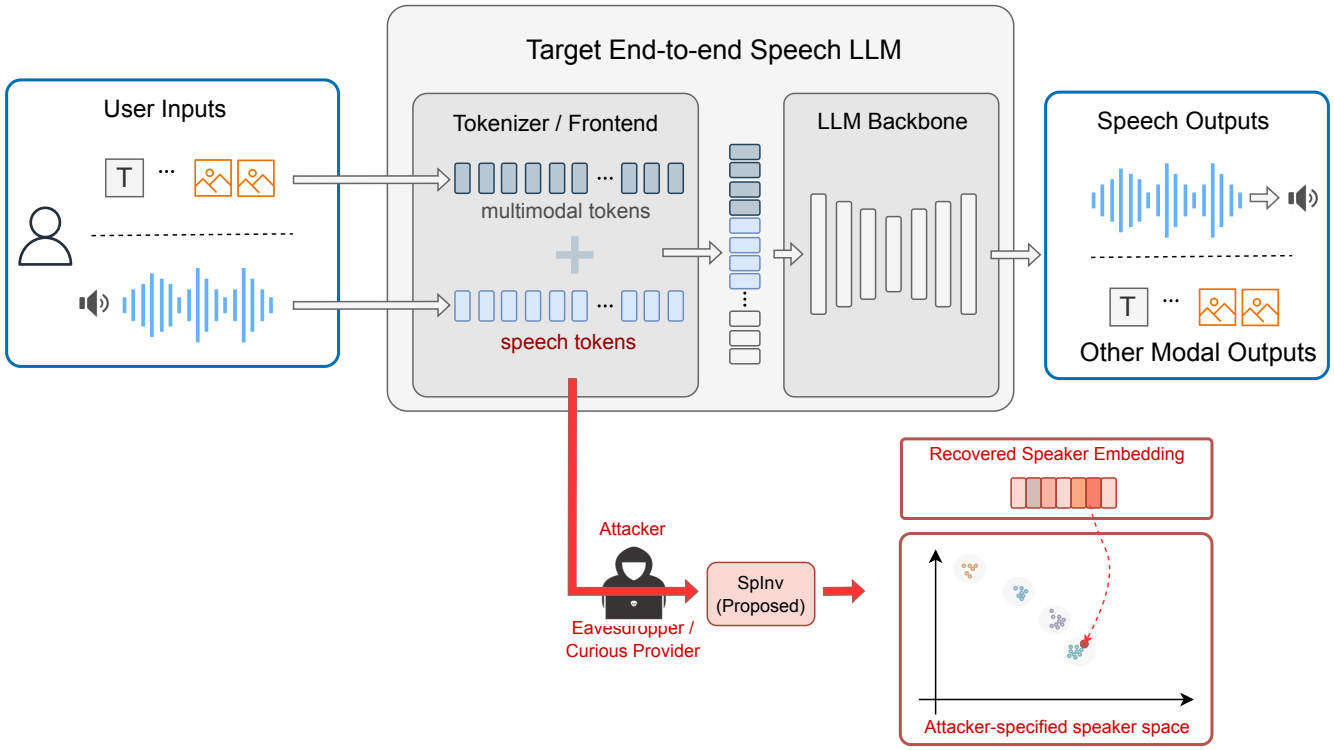

Fig. 1. Attack scenario. User speech is converted into speech tokens before large-model inference. An eavesdropper or curious provider observing the exposed speech interface uses SpInv to recover an embedding in the space of an attacker-specified speaker encoder.

Representative systems such as SpeechGPT [2], Spectron [3], and PSLM [4] connect speech encoders, language models, and speech-generation modules, retaining structural similarities to cascaded ASR–LLM–TTS pipelines.

Recent systems increasingly retain speech-specific representations throughout a unified inference pipeline. By avoiding repeated conversion between speech and text, speech-token modeling preserves paralinguistic information and supports lower-latency, full-duplex spoken dialogue, which better matches the interactive requirements of speech language models.

Moshi [5] is a real-time, full-duplex speech–text model whose Mimi codec maps 24-kHz audio to one semantic and seven residual acoustic token streams at 12.5 frames/s. Higgs3 [6] is a 4B autoregressive conversational TTS model using eight codebooks at 25 frames/s; its tokenizer is also adopted by the diffusion-based OmniVoice [7]. Kimi-Audio [8] combines a 12.5-Hz semantic-token stream with time-aligned continuous Whisper features. Qwen3-Omni [9] instead uses an Audio Transformer to emit continuous speech representations at 12.5 Hz. Together, these models cover residual-codebook, hybrid discrete–continuous, and continuous interfaces; AuB handles all three within one voiceprint-leakage attack pipeline.

B. Speaker Representation Learning

Speaker recognition has evolved from statistical pipelines to deep speaker embeddings. Early methods used GMM-UBM models, joint factor analysis, and i-vectors [10]–[12]. Subsequent speaker-verification systems learned discriminative embeddings from large speech corpora [13]–[17], with recent work also exploring self-supervised speaker representation learning [18], [19]. ECAPA-TDNN improves TDNN-based verification through channel attention and feature aggregation [20]; CAM++ uses context-aware masking for efficient speaker modeling [21]; and ERes2Net combines local and global multi-scale features [22]. In our threat model, the adversary selects a pretrained speaker encoder to define the target space into which SpInv recovers embeddings.

C. Model Inversion and Audio Privacy Attacks

Model Inversion Attacks (MIAs) infer sensitive information about model inputs or training data from outputs and intermediate representations. They have been extensively studied in vision, where an adversary may reconstruct representative inputs or identity-related features [1], [23]–[25]. Related studies in natural language processing have shown that continuous text embeddings can retain enough information to recover substantial properties of their source text [26], [27]. These findings demonstrate that replacing raw inputs with learned representations does not necessarily remove private information.

Speech and audio systems expose several distinct privacy surfaces. Prior work has introduced inversion attacks against automatic speaker-recognition systems [28] and ex-

amined whether speaker identity remains inferable from representations produced by speaker-anonymization mechanisms [29]. Membership-inference studies further investigate whether speech representations reveal the participation of particular users or sensitive attributes in model training [30]. Other attacks exploit released speaker scores to support impersonation [31], while recent evaluations of audio language models consider whether their outputs disclose sensitive information contained in the audio they process [32]. Our work instead targets speech tokens exposed between the speech frontend and the downstream backbone, using a partial token sequence to recover an embedding in the space of an attacker-specified speaker encoder. Unlike waveform-oriented inversion or attacks against speaker-recognition systems, SpInv does not reconstruct the original speech signal or attack the speaker encoder; instead, it recovers an embedding in an attacker-specified speaker space.

III. Method

A. Threat Model

We formalize the attack surface, the adversary's objective, and the knowledge available to the adversary.

a) Target model: Let x denote an arbitrary user speech waveform. The target system is an end-to-end speech language model whose speech frontend converts user audio into speech tokens before downstream inference. Let T denote the tokenizer in this frontend, which converts x into a time-indexed speech-token sequence

$$\begin{aligned} S = T(x) = [\mathbf{s}_1, \ldots, \mathbf{s}_N], \\ \mathbf{s}_n = (s_n^{(1)}, \ldots, s_n^{(Q)}). \end{aligned} \tag{1}$$

Here, N is the sequence length and Q is the number of codebooks. The q-th codebook has size M_q, and $s_n^{(q)} \in \{1, \ldots, M_q\}$ is its token ID at frame n. Some frontends additionally expose time-aligned continuous features $A = [\mathbf{a}_1, \ldots, \mathbf{a}_N]$ alongside the token stream or in place of it. We denote the complete exposed speech representation by $R(x) = (S, A)$, with any unavailable component omitted. We consider deployments in which these frontend outputs are accessible to an intermediate service or transmitted to the downstream model.

b) Adversary goal: The adversary is an eavesdropper observing the exposed speech interface or a curious service provider with access to it during inference, consistent with speech-privacy threat models that assume access to intermediate representations [30], [32]. Let F be a frozen speaker encoder selected by the attacker that maps a waveform to a target speaker embedding

$$\mathbf{z} = F(x) \in \mathbb{R}^{d_z}, \tag{2}$$

where d_z is the embedding dimension. By selecting F, the attacker specifies the target speaker-embedding space to be recovered. Rather than reconstructing the original waveform or recovering the target model's parameters, the adversary observes a randomly cropped contiguous segment $\widetilde{R}$ of $R(x)$ and seeks to recover $\mathbf{z}$ without access to the victim's waveform at attack time. A recovery model G_θ with parameters θ produces $\hat{\mathbf{z}} = G_\theta(\widetilde{R}) \in \mathbb{R}^{d_z}$. We formulate the Speaker Inversion Attack as

$$\theta^* = \arg\min_\theta \ \mathcal{L}\big(G_\theta(\widetilde{R}), \mathbf{z}\big), \tag{3}$$

where $\mathcal{L}$ measures the discrepancy between the recovered and target embeddings in the speaker space defined by F, and θ^* denotes the optimized recovery-model parameters.

c) Adversary knowledge and capabilities: The adversary is assumed to have three capabilities. First, it has query access to the target speech frontend T: it can submit an utterance and observe the corresponding frontend output $R(x)$, but it does not access the internal parameters of the target model. Second, for a token-based interface, it can observe the number of codebooks Q and their vocabulary sizes $\{M_q\}_{q=1}^{Q}$; the learned codebook embedding values are neither known nor required. Third, it can obtain a public auxiliary speech dataset $\mathcal{D}$ and use the queried frontend outputs to train the attack model. The adversary selects and has access to the frozen speaker encoder F; this choice determines the desired speaker-embedding space into which G_θ must recover embeddings. In our experiments, F is instantiated by a publicly available pretrained speaker encoder.

B. Overview of SpInv

As illustrated in Fig. 2, SpInv encompasses both a training phase and an attack phase. The training phase learns a recovery model G_θ from the outputs of the target speech frontend through a two-stage strategy that combines distillation pretraining and discriminative fine-tuning. In the attack phase, the adversary intercepts only a slice of exposed speech tokens from the victim's frontend stream; the trained AuB model and projection head then recover a speaker embedding $\hat{\mathbf{z}}$ in the selected speaker space directly from this token slice, without requiring the original waveform.

C. AuB Model Structure

AuB is the trainable speaker information extractor used by SpInv. It comprises a codebook embedding layer, a feature fusion layer, and a BERT backbone. For codebook q with vocabulary size M_q, AuB maintains a trainable lookup table $\mathbf{E}^{(q)} \in \mathbb{R}^{M_q \times d}$. The token IDs at frame n are mapped to the shared embedding space and added across codebooks:

$$\mathbf{e}_n = \sum_{q=1}^{Q} \mathbf{E}^{(q)}[s_n^{(q)}], \qquad \mathbf{e}_n \in \mathbb{R}^d. \tag{4}$$

The lookup tables belong to AuB and are learned independently of the unknown codebook embeddings used by the target model.

Since certain target frontends additionally expose the time-aligned continuous feature $\mathbf{a}_n$ at each frame as supplementary acoustic information, AuB incorporates a feature fusion layer for compatibility. An MLP projects this feature to the shared

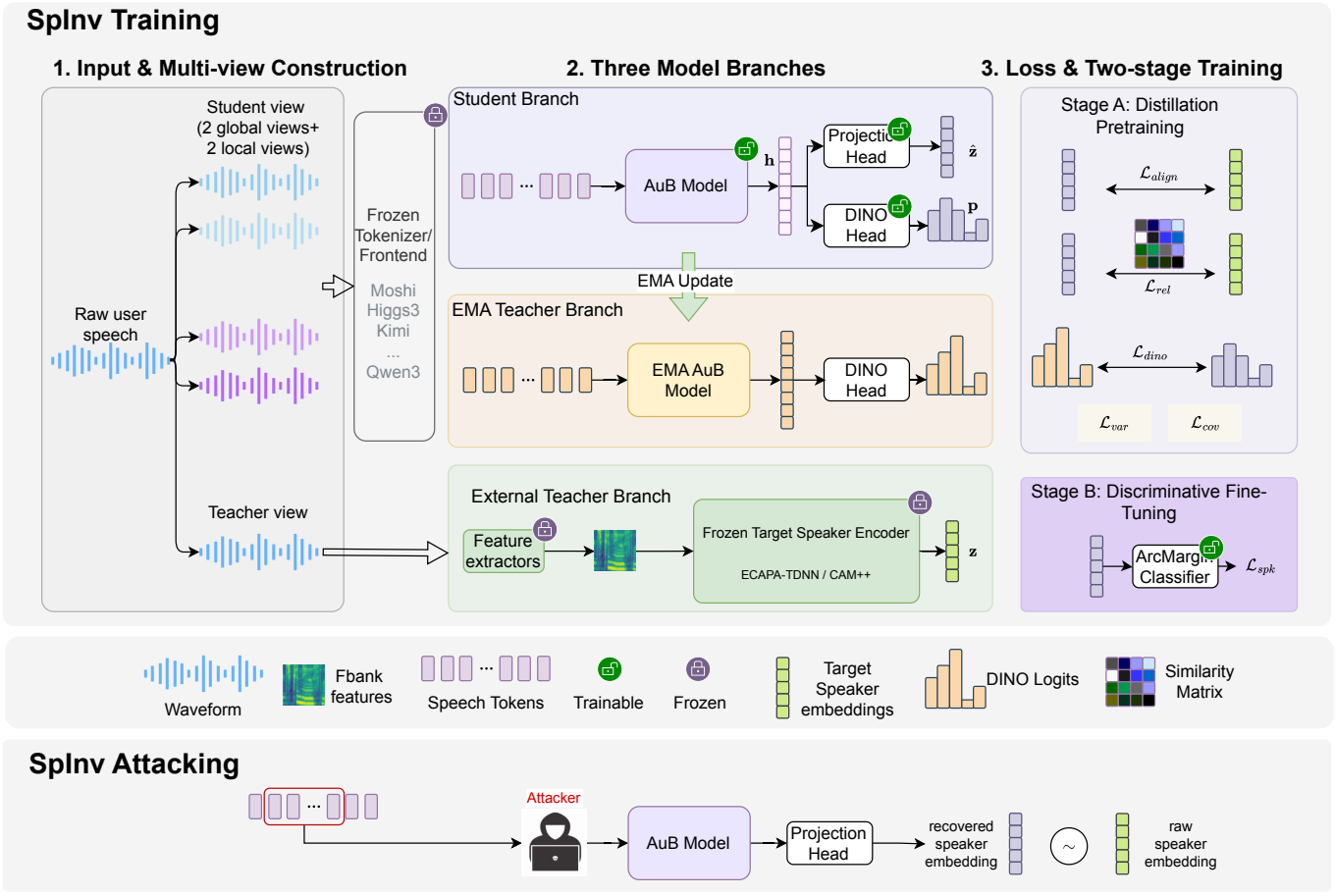

Fig. 2. Overview of SpInv. The training phase learns a recovery model from frontend outputs via two-stage optimization; the attack phase recovers a speaker embedding from an intercepted token slice.

embedding space as $\mathbf{u}_n = \mathrm{MLP}(\mathbf{a}_n) \in \mathbb{R}^d$. The fused representation at frame n is obtained by element-wise addition:

$$\mathbf{r}_n = \mathbf{e}_n + \mathbf{u}_n. \tag{5}$$

This design accommodates the native exposed outputs of all target frontends within a single architecture.

Similar to ViT [33], AuB prepends a learnable CLS token to $[\mathbf{r}_1, \ldots, \mathbf{r}_L]$ and processes the sequence with a BERT backbone [34], where L is the length of the input segment. The CLS output $\mathbf{h}$ is converted into a fixed 512-dimensional AuB output $\mathbf{o} \in \mathbb{R}^{d_a}$, where $d_a = 512$. The projection head is a two-layer MLP with LayerNorm [35] and GELU [36] that maps $\mathbf{o}$ to the recovered embedding $\hat{\mathbf{z}} \in \mathbb{R}^{d_z}$. Accordingly, the recovery model G_θ in (3) comprises AuB and the projection head, with $\theta = (\theta_A, \theta_P)$ denoting their respective parameters. During training, G_θ also exposes $\mathbf{h}$ and $\mathbf{o}$ for auxiliary objectives. A separate DINO head D_{ϕ_D} uses a two-layer MLP projector with GELU, followed by bottleneck LayerNorm, ℓ_2 normalization, and a bias-free prototype layer to produce logits $\boldsymbol{\ell} = D_{\phi_D}(\mathbf{o})$. The DINO head is optimized only as an auxiliary training component and is discarded after training; it is therefore not part of G_θ or the final recovery parameters θ^*. AuB, the projection head, and the auxiliary DINO head are randomly initialized and trained jointly.

D. SpInv Training and Attack

a) Training phase: As shown in the training phase of Fig. 2, SpInv organizes three parallel branches. All five views are constructed at the waveform level for each utterance: two local views $\mathcal{V}_s = \{l_1, l_2\}$, two global views $\mathcal{V}_g = \{g_1, g_2\}$, and one teacher view x_i^t. The student model receives only the two local views through the frozen target frontend R and G_θ, producing recovered embeddings $\hat{\mathbf{z}}$, CLS representations $\mathbf{h}$, AuB outputs $\mathbf{o}$, and student logits from D_{ϕ_D}. The EMA teacher model, comprising EMA copies of AuB and the DINO head, receives only the two global views and generates self-distillation targets. The frozen teacher model F receives only the teacher view and produces the target embedding $\mathbf{z}_i = F(x_i^t)$. The loss block in Fig. 2 shows that L_{align} and L_{rel} operate on embeddings recovered from the local views, L_{dino} matches student distributions from the local views against EMA teacher targets from the global views, and L_{var} and L_{cov} regularize representations from the local views. In Stage B, an ArcMargin classifier adds speaker-discriminative supervision on the CLS representations from the local views.

Stage A: Distillation pretraining. Stage A combines alignment to the target speaker space, relational matching of batch geometry, and EMA-based self-distillation across views. Variance and covariance regularization following VICReg [37] prevent representational collapse. Its objective is

$$L_A = \lambda_1 L_{\text{align}} + \lambda_2 L_{\text{rel}} + \lambda_3 L_{\text{dino}} + \lambda_4 L_{\text{var}} + \lambda_5 L_{\text{cov}}, \quad (6)$$

where $\lambda_j \geq 0$ are loss weights.

For a mini-batch of B utterances, let $V_s = |\mathcal{V}_s| = 2$, let $\hat{\mathbf{z}}_{i,v}$ be the recovered embedding of utterance i under local view $v \in \mathcal{V}_s$, and let $\mathbf{z}_i$ be the target embedding obtained from its teacher view. The alignment loss is

$$L_{\text{align}} = \frac{1}{BV_s} \sum_{i=1}^{B} \sum_{v \in \mathcal{V}_s} \big(1 - \cos(\hat{\mathbf{z}}_{i,v}, \mathbf{z}_i)\big). \quad (7)$$

The relational loss aligns pairwise speaker geometry within the batch:

$$L_{\text{rel}} = \frac{1}{V_s B(B-1)} \sum_{v \in \mathcal{V}_s} \sum_{i \neq j} \big(\cos(\hat{\mathbf{z}}_{i,v}, \hat{\mathbf{z}}_{j,v}) - \cos(\mathbf{z}_i, \mathbf{z}_j)\big)^2. \quad (8)$$

For self-distillation, the valid teacher–student pairs form the Cartesian product $\mathcal{P} = \mathcal{V}_g \times \mathcal{V}_s$. The EMA teacher produces centered and sharpened prototype distributions $q_{i,u}$ from global view $u \in \mathcal{V}_g$, while the student produces prototype distributions $p_{i,v}$ from local view $v \in \mathcal{V}_s$. The DINO loss [38] is

$$L_{\text{dino}} = -\frac{1}{B|\mathcal{P}|} \sum_{(u,v) \in \mathcal{P}} \sum_{i=1}^{B} \sum_{k=1}^{K} \text{sg}(q_{i,u,k}) \log p_{i,v,k}, \quad (9)$$

where K is the number of prototypes and $\text{sg}(\cdot)$ denotes stop-gradient.

Let $H^{(v)} \in \mathbb{R}^{B \times d_h}$ contain the student representations used for regularization under view v. With $\gamma > 0$ denoting the target standard deviation and $\epsilon > 0$ ensuring numerical stability, the view-averaged anti-collapse terms are

$$L_{\text{var}} = \frac{1}{V_s d_h} \sum_{v \in \mathcal{V}_s} \sum_{j=1}^{d_h} \max\big(0, \gamma - \sqrt{\text{Var}(H^{(v)}_{:,j}) + \epsilon}\big), \quad (10)$$

$$L_{\text{cov}} = \frac{1}{V_s d_h} \sum_{v \in \mathcal{V}_s} \sum_{j \neq j'} \big(\text{Cov}(H^{(v)})_{j,j'}\big)^2. \quad (11)$$

Stage B: Discriminative fine-tuning. Stage B initializes G_θ and the auxiliary DINO head from the selected Stage A checkpoint and adds speaker-discriminative supervision. An ArcMargin classifier [39] is applied to $\mathbf{h}$, while lower-weight versions of the Stage A losses preserve the learned target-space structure:

$$L_B = \mu_1 L_{\text{spk}} + \mu_2 L_{\text{align}} + \mu_3 L_{\text{rel}} + \mu_4 L_{\text{dino}} + \mu_5 L_{\text{var}} + \mu_6 L_{\text{cov}}, \quad (12)$$

where $\mu_j \geq 0$ are loss weights. Let C_{spk} be the number of speakers in the auxiliary training set, with $c \in \{1, \ldots, C_{\text{spk}}\}$. For normalized $\bar{\mathbf{h}}_{i,v}$ and classifier weight $\bar{\mathbf{w}}_c$, define $\alpha_{i,v,c} = \arccos(\bar{\mathbf{h}}_{i,v}^\top \bar{\mathbf{w}}_c)$ and the ArcMargin logit $\beta_{i,v,c} = s \cos(\alpha_{i,v,c} + m\mathbb{I}[c = y_i])$. The speaker-classification loss over the local views is

$$L_{\text{spk}} = -\frac{1}{BV_s} \sum_{i=1}^{B} \sum_{v \in \mathcal{V}_s} \log \frac{e^{\beta_{i,v,y_i}}}{\sum_{c=1}^{C_{\text{spk}}} e^{\beta_{i,v,c}}}, \quad (13)$$

where y_i is the speaker label, m is the angular margin, and s is the logit scale. Stage B retains the same assignment as Stage A: two local views to the student model, two global views to the EMA teacher model, and one teacher view to the teacher model.

b) Attack phase: As illustrated in the attack phase of Fig. 2, the adversary does not access the victim's waveform but only intercepts a contiguous slice of exposed speech tokens (and any accompanying interface features) from the victim's frontend output stream. This token slice is fed directly to the trained recovery model G_θ, comprising AuB and the projection head, to yield $\hat{\mathbf{z}}$ in the speaker space specified by F. The auxiliary DINO head and ArcMargin classifier are not used during the attack. As demonstrated experimentally, the recovered speaker embedding $\hat{\mathbf{z}}$ has high cosine similarity to the raw speaker embedding $\mathbf{z} = F(x)$ extracted by F when the waveform is available for evaluation.

IV. Experiments

A. Experimental Setup

a) Target models: We evaluate SpInv against four representative speech-model frontends with their native exposed outputs. Moshi [5] employs the Mimi neural audio codec, which exposes one semantic and seven residual acoustic token streams at 12.5 frames/s. Higgs3 [6] uses an eight-codebook tokenizer operating at 25 frames/s; this tokenizer is also used as the audio frontend in OmniVoice [7]. Kimi-Audio [8] outputs a semantic-token stream together with time-aligned continuous Whisper features, which provide complementary acoustic information. Qwen3-Omni [9] uses a continuous Audio Transformer frontend that emits frame-level feature vectors at 12.5 Hz. The first three models expose discrete speech-token interfaces and constitute the core focus of this study; Qwen3-Omni is additionally evaluated to demonstrate that SpInv also applies to purely continuous frontend features, broadening the evaluation coverage. SpInv uses the exposed output of each frontend directly. For each frontend, input waveforms are resampled to its native sampling rate; all frontend parameters remain frozen throughout attack training.

b) Datasets: We use the VoxCeleb corpora, which are widely adopted benchmarks for speaker recognition research. VoxCeleb1 [40] contains over 150,000 utterances from 1,251 speakers, totaling approximately 352 hours of YouTube speech. Under its standard verification partition, the development set contains 1,211 speakers and the test set contains 40 speakers, with no speaker identity overlap. VoxCeleb2 [17] is substantially larger, comprising over 1.1 million utterances, totaling roughly 2,442 hours, from 6,112 speakers. Its standard partition contains 5,994 development speakers and 118 test speakers, which are also identity-disjoint. Both corpora cover

diverse acoustic conditions, speaking styles, and demographic backgrounds.

We follow the standard VoxCeleb1 and VoxCeleb2 evaluation protocols. Privacy leakage validation uses the official VoxCeleb1 verification split [40]. For speaker embedding recovery, SpInv is trained on VoxCeleb2-dev and evaluated on the speaker-disjoint VoxCeleb2-test and VoxCeleb1 sets. The speakers in VoxCeleb1 and VoxCeleb2-dev do not overlap. Unless stated otherwise, the subsequent ablations use the Moshi tokenizer with VoxCeleb2-dev $\rightarrow$ VoxCeleb2-test.

c) Evaluation method: The frozen teacher model extracts target speaker embeddings from the teacher view, while the student model receives only target-frontend outputs derived from the local views. Let $\mathbf{z}_i$ be the target embedding extracted by the frozen speaker encoder F and $\hat{\mathbf{z}}_i$ be the embedding recovered by SpInv. CosSim measures their average cosine similarity:

$$\mathrm{CosSim} = \frac{1}{N}\sum_{i=1}^{N}\frac{\hat{\mathbf{z}}_i^\top \mathbf{z}_i}{\|\hat{\mathbf{z}}_i\|_2\|\mathbf{z}_i\|_2}. \tag{14}$$

We also report KL after converting the two embeddings into distributions over dimensions, $P_i = \mathrm{softmax}(\mathbf{z}_i)$ and $\hat{P}_i = \mathrm{softmax}(\hat{\mathbf{z}}_i)$:

$$\mathrm{KL} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k} P_{i,k}\log\frac{P_{i,k}+\epsilon}{\hat{P}_{i,k}+\epsilon}, \tag{15}$$

where ϵ is a small constant. Accuracy (Acc.), equal error rate (EER), and normalized minDCF measure speaker-verification performance. For minDCF, we use $P_{\mathrm{target}} = 0.01$, $C_{\mathrm{miss}} = 1$, and $C_{\mathrm{fa}} = 1$. The reported threshold is the corresponding optimal cosine-score decision boundary used to compute Acc.

d) Target speaker space: The target speaker space is defined by the frozen encoder F selected by the attacker, as formalized in (2). In the main experiments, we adopt ECAPA-TDNN [20] with embedding dimensionality $d_z = 192$ as the default target space. To assess the robustness of SpInv across different speaker representations, we also evaluate three alternative target spaces: ERes2Net [22] ($d_z = 192$), which captures multi-scale speaker features, and CAM++ [21] at two embedding dimensionalities ($d_z = 192$ and $d_z = 512$), which employs context-aware masking for efficient speaker modeling. All target encoders are publicly available, pretrained, and kept frozen during attack training.

e) Default hyperparameter configuration: The AuB codebook embedding layer uses the same number of codebooks and vocabulary sizes as observed in the target frontend. The feature fusion layer uses a one-layer MLP. The projection and DINO heads follow the structures described in Section III-C. AuB uses a BERT hidden dimension of $d_h = 384$ and produces a fixed-dimensional output with $d_a = 512$. The projection head maps this 512-dimensional output to d_z, the dimensionality of the selected target speaker encoder, such as $d_z = 192$ for the default ECAPA-TDNN. The complete AuB model is trained from random initialization using Adam with a batch size of 96 (which occupies approximately 20 GB of GPU memory), while the target frontend and speaker encoder remain frozen. Stage A runs for 60 epochs with a learning rate of 4×10^{-4} and weights $\lambda_1 = 1.0$, $\lambda_2 = 0.5$, $\lambda_3 = 1.0$, $\lambda_4 = 25.0$, and $\lambda_5 = 1.0$ for L_{align}, L_{rel}, L_{dino}, L_{var}, and L_{cov}, respectively. Stage B runs for 40 epochs with a learning rate of 1×10^{-4} and weights $\mu_1 = 0.2$, $\mu_2 = 0.8$, $\mu_3 = 0.4$, $\mu_4 = 0.4$, $\mu_5 = 10.0$, and $\mu_6 = 0.5$ for L_{spk}, L_{align}, L_{rel}, L_{dino}, L_{var}, and L_{cov}. The ArcMargin margin is linearly increased from 0 to 0.2 between epochs 10 and 20 of Stage B. The DINO center momentum is 0.9, and the EMA momentum follows a cosine schedule from 0.996 to 0.9995.

TABLE I
SPEAKER-INFORMATION LEAKAGE ON THE SPEAKER-DISJOINT VOXCELEB1 VERIFICATION SPLIT.

Target model	EER ↓	minDCF ↓
Moshi	0.0160	0.0512
Higgs3	0.0066	0.0132
Kimi-Audio	0.0034	0.0171
Qwen3-Omni	0.0236	0.0836

TABLE II
SPEAKER EMBEDDING RECOVERY FROM VOXCELEB2-DEV TO VOXCELEB2-TEST.

Target model	CosSim ↑	KL ↓	Acc ↑	EER ↓	minDCF ↓	Thres.
Moshi	0.7136	0.3346	0.9701	0.0305	0.0885	0.3287
Higgs3	0.7527	0.2858	0.9828	0.0174	0.0627	0.3870
Kimi-Audio	0.7536	0.2919	0.9825	0.0178	0.0723	0.4215
Qwen3-Omni	0.6683	0.3257	0.9624	0.0378	0.1124	0.3269

B. Privacy Leakage Validation

Before evaluating inversion, we determine whether the exposed speech interfaces retain speaker-discriminative information. We train AuB with speaker labels on the VoxCeleb1 development partition and evaluate its representations on the speaker-disjoint VoxCeleb1 verification test partition. Random three-second cropping is the only data augmentation. Table I reports this diagnostic experiment for all four target models.

All four exposed interfaces retain strong speaker-discriminative information under the speaker-disjoint protocol. Kimi-Audio achieves the lowest EER of 0.0034, followed by Higgs3 at 0.0066 and Moshi at 0.0160, while Qwen3-Omni also exhibits substantial leakage with an EER of 0.0236. These results confirm that the frontend representations preserve speaker information that generalizes to unseen identities.

C. Speaker Embedding Recovery

We next evaluate whether SpInv can recover embeddings in the frozen ECAPA-TDNN speaker space for previously unseen speakers. By default, the attack observes three seconds of exposed frontend output. Table II reports results on VoxCeleb2-test, and Table III reports results on VoxCeleb1; both use VoxCeleb2-dev for attack training.

Across both speaker-disjoint evaluation sets, Higgs3 and Kimi-Audio provide the strongest overall recovery, followed by Moshi, while Qwen3-Omni is the most difficult frontend to

TABLE III
SPEAKER EMBEDDING RECOVERY FROM VOXCELEB2-DEV TO VOXCELEB1.

Target model	CosSim ↑	KL ↓	Acc ↑	EER ↓	minDCF ↓	Thres.
Moshi	0.7037	0.4257	0.9395	0.0606	0.1245	0.2833
Higgs3	0.7060	0.3540	0.9513	0.0489	0.1016	0.3208
Kimi-Audio	0.7359	0.3395	0.9449	0.0554	0.0991	0.3422
Qwen3-Omni	0.6594	0.3991	0.9219	0.0783	0.1335	0.2240

TABLE IV
EFFECT OF ATTACK-TIME INPUT DURATION USING MOSHI (VOXCELEB2-DEV → VOXCELEB2-TEST).

Duration	CosSim ↑	KL ↓	Acc ↑	EER ↓	minDCF ↓	Thres.
1 s	0.4887	0.4353	0.8656	0.1346	0.6670	0.1868
2 s	0.6677	0.3537	0.9559	0.0444	0.4050	0.2697
3 s	0.7136	0.3346	0.9701	0.0305	0.0885	0.3287
5 s	0.7642	0.3176	0.9849	0.0153	0.0878	0.3694
8 s	0.7949	0.3131	0.9892	0.0110	0.0707	0.3723
10 s	0.8065	0.3116	0.9878	0.0124	0.0757	0.4132

attack. All four frontends perform worse on VoxCeleb1 than on VoxCeleb2-test. Because SpInv is trained on VoxCeleb2-dev, this consistent decline may result from a slight distribution mismatch between the two evaluation datasets, while the relative performance trend remains stable.

D. Ablation Study

We examine the effects of attack-time input duration and the target speaker space using Moshi with VoxCeleb2-dev → VoxCeleb2-test.

a) Input duration: We vary the observed segment duration from 1 to 10 seconds while using ECAPA-TDNN as the target speaker encoder. Table IV reports how the amount of exposed speech affects embedding recovery.

Shorter observations reduce speaker-embedding recovery performance, with the largest degradation occurring at one second. Starting from three seconds, the attack achieves substantially stronger recovery. Longer observations generally improve CosSim, although the verification gains become smaller and mildly non-monotonic beyond eight seconds, indicating that the attack approaches saturation.

b) Target speaker space: We evaluate recovery into several attacker-specified speaker-encoder spaces instantiated with publicly available models, including ECAPA-TDNN, ERes2Net [22], and CAM++ [21]. Table V reports the results for each target space.

SpInv achieves effective recovery across all four target speaker spaces. CAM++ with 192-dimensional embeddings obtains the highest CosSim of 0.7658, whereas its 512-dimensional variant achieves the lowest EER of 0.0135 and the highest Acc. of 0.9867. These results indicate that the attack is not restricted to a specific speaker encoder or embedding dimensionality.

TABLE V
EFFECT OF THE TARGET SPEAKER SPACE USING MOSHI (VOXCELEB2-DEV → VOXCELEB2-TEST).

Speaker encoder	Dim	CosSim ↑	Acc ↑	EER ↓	minDCF ↓	Thres.
ECAPA-TDNN	192	0.7136	0.9701	0.0305	0.0885	0.3287
ERes2Net	192	0.7562	0.9559	0.0443	0.0882	0.3389
CAM++	192	0.7658	0.9811	0.0191	0.0957	0.3074
CAM++	512	0.7167	0.9867	0.0135	0.1044	0.3040

V. CONCLUSION

This paper investigates whether speech tokens exposed by end-to-end speech language models leak sufficient voiceprint information to recover speaker embeddings. We formulate this risk as a speaker inversion attack and propose SpInv, a two-stage method that uses AuB to construct and aggregate discrete or continuous frontend representations and recover embeddings in an attacker-specified speaker space. We evaluate SpInv against Moshi, Higgs3, Kimi-Audio, and Qwen3-Omni using speaker-disjoint protocols on VoxCeleb2-test and VoxCeleb1. The results show that exposed frontend representations retain substantial speaker information: with only three seconds of observation, SpInv achieves strong recovery on unseen speakers, and the three token-based frontends generally reach cosine similarities above 0.70.

This study has two main limitations. First, it focuses on recovering speaker embeddings rather than reconstructing waveforms that reproduce a victim's acoustic characteristics. Second, the experiments are limited to the VoxCeleb corpora rather than a broader range of unconstrained in-the-wild speech datasets. Future work will investigate privacy-preserving tokenizers that better suppress voiceprint information while retaining the content needed by downstream speech-language tasks. We also plan to study stronger attacks that recover waveforms from exposed speech representations.

VI. ETHICAL STATEMENT

This work is intended to identify and quantify voiceprint-privacy risks in exposed speech representations so that safer speech interfaces can be developed. All experiments use publicly available research datasets and pretrained models, and results are reported only in aggregate. We do not attempt to identify, contact, or impersonate individuals represented in the evaluation data, nor do we reconstruct their speech waveforms. Nevertheless, speaker inversion could be misused for unauthorized profiling or identity inference. Any release or application of such techniques should follow applicable dataset licenses, consent requirements, and data-protection regulations, and should incorporate access controls that prevent attacks on non-consenting users. Our findings are presented to motivate privacy-preserving tokenizers and stronger safeguards for deployed speech-language systems.

REFERENCES

[1] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 253–261.

[2] D. Zhang, S. Li, X. Zhang, J. Zhan, P. Wang, Y. Zhou, and X. Qiu, "SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities," in *Findings Assoc. Comput. Linguist. (EMNLP)*, 2023, pp. 15 757–15 773.

[3] E. Nachmani, A. Levkovitch, R. Hirsch, J. Salazar, C. Asawaroengchai, S. Mariooryad, E. Rivlin, R. J. Skerry-Ryan, and M. Tadmor Ramanovich, "Spoken question answering and speech continuation using spectrogram-powered LLM," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.

[4] K. Mitsui, K. Mitsuda, T. Wakatsuki, Y. Hono, and K. Sawada, "PSLM: Parallel generation of text and speech with LLMs for low-latency spoken dialogue systems," in *Findings Assoc. Comput. Linguist. (EMNLP)*, 2024, pp. 2692–2700.

[5] A. Défossez, L. Mazaré, M. Orsini, A. Royer, P. Pérez, H. Jégou, E. Grave, and N. Zeghidour, "Moshi: A speech-text foundation model for real-time dialogue," *arXiv preprint arXiv:2410.00037*, 2024.

[6] Boson AI, "Higgs TTS 3: Conversational speech for voice AI from Boson AI," https://huggingface.co/bosonai/higgs-tts-3-4b, 2026.

[7] H. Zhu, L. Ye, W. Kang, Z. Yao, L. Guo, F. Kuang, Z. Han, W. Zhuang, L. Lin, and D. Povey, "OmniVoice: Towards omnilingual zero-shot text-to-speech with diffusion language models," *arXiv preprint arXiv:2604.00688*, 2026.

[8] Kimi Team, D. Ding, Z. Ju, Y. Leng, S. Liu, T. Liu, Z. Shang, K. Shen, W. Song, X. Tan *et al.*, "Kimi-Audio technical report," *arXiv preprint arXiv:2504.18425*, 2025.

[9] J. Xu, Z. Guo, H. Hu, Y. Chu, X. Wang, J. He, Y. Wang, X. Shi, T. He, X. Zhu *et al.*, "Qwen3-Omni technical report," *arXiv preprint arXiv:2509.17765*, 2025.

[10] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digit. Signal Process.*, vol. 10, no. 1–3, pp. 19–41, 2000.

[11] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Trans. Audio Speech Lang. Process.*, vol. 15, no. 4, pp. 1435–1447, 2007.

[12] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, no. 4, pp. 788–798, 2011.

[13] E. Variani, X. Lei, E. McDermott, I. Lopez Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2014, pp. 4052–4056.

[14] G. Heigold, I. Lopez-Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2016, pp. 5115–5119.

[15] L. Wan, Q. Wang, A. Papir, and I. Lopez Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2018, pp. 4879–4883.

[16] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2018, pp. 5329–5333.

[17] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (Interspeech)*, 2018, pp. 1086–1090.

[18] Y. Chen, S. Zheng, H. Wang, L. Cheng, and Q. Chen, "Pushing the limits of self-supervised speaker verification using regularized distillation framework," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2023, pp. 1–5.

[19] Y. Chen, S. Zheng, H. Wang, L. Cheng, Q. Chen, C. Deng, S. Zhang, and W. Wang, "Self-distillation prototypes network: Learning robust speaker representations without supervision," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2025, pp. 1–5.

[20] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (Interspeech)*, 2020, pp. 3830–3834.

[21] H. Wang, S. Zheng, Y. Chen, L. Cheng, and Q. Chen, "CAM++: A fast and efficient network for speaker verification using context-aware masking," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (Interspeech)*, 2023, pp. 5301–5305.

[22] Y. Chen, S. Zheng, H. Wang, L. Cheng, Q. Chen, and J. Qi, "An enhanced Res2Net with local and global feature fusion for speaker verification," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (Interspeech)*, 2023, pp. 2228–2232.

[23] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2015, pp. 1322–1333.

[24] Y. Lu, S. Wang, G. Zhu, Z. Zhang, and J. Huang, "FGMIA: Feature-guided model inversion attacks against face recognition models," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 8465–8480, 2025.

[25] O. Li, Y. Hao, Z. Wang, B. Zhu, S. Wang, Z. Zhang, and F. Feng, "Model inversion attacks through target-specific conditional diffusion models," *arXiv preprint arXiv:2407.11424*, 2024.

[26] J. X. Morris, V. Kuleshov, V. Shmatikov, and A. M. Rush, "Text embeddings reveal (almost) as much as text," in *Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP)*, 2023, pp. 12 448–12 460.

[27] Y. Chen, H. Lent, and J. Bjerva, "Text embedding inversion security for multilingual language models," in *Proc. Annu. Meet. Assoc. Comput. Linguist. (ACL)*, 2024, pp. 7808–7827.

[28] K. Pizzi, F. Boenisch, U. Sahin, and K. Böttinger, "Introducing model inversion attacks on automatic speaker recognition," in *Proc. Symp. Secur. Priv. Speech Commun. (SPSC)*, 2022, pp. 11–16.

[29] L. A. Bauer, W. Bao, M. Jadhav, and V. Bindschaedler, "Inference attacks for x-vector speaker anonymization," in *Proc. IEEE Secur. Priv. Workshops (SPW)*, 2025, pp. 152–159.

[30] E. Tsaprazlis, T. Lertpetchpun, T. Feng, S. P. Karimireddy, and S. Narayanan, "VoxGuard: Evaluating user and attribute privacy in speech via membership inference attacks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2026, pp. 19 042–19 046.

[31] C. Hwang, S. Kim, Y. K. Tan, T. Liu, S. Paik, D. Kim, M. Soumik, K. M. M. Aung, and J. H. Seo, "Scores know Bob's voice: Speaker impersonation attack," *arXiv preprint arXiv:2603.02781*, 2026.

[32] J. Wang, K. Luo, L. Lin, W. Wang, Y. Chen, M. Aloqaily, X. Tang, Z. Zhou, K. Wang, L. Sun, and Q. Wen, "HearSay benchmark: Do audio LLMs leak what they hear?" in *Findings Assoc. Comput. Linguist. (ACL)*, 2026, pp. 19 312–19 331.

[33] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. North Am. Chapter Assoc. Comput. Linguist.: Hum. Lang. Technol. (NAACL-HLT)*, 2019, pp. 4171–4186.

[35] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[36] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016.

[37] A. Bardes, J. Ponce, and Y. LeCun, "VICReg: Variance-invariance-covariance regularization for self-supervised learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022.

[38] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 9650–9660.

[39] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 4690–4699.

[40] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (Interspeech)*, 2017, pp. 2616–2620.