VENOMAVE: Targeted Poisoning Against Speech Recognition

Hojjat Aghakhani[1], Lea Schönherr[2], Thorsten Eisenhofer[2], Dorothea Kolossa[2],
Thorsten Holz[2], Christopher Kruegel[1], and Giovanni Vigna[1]

[1]University of California, Santa Barbara
{hojjat, chris, vigna}@cs.ucsb.edu
[2]Ruhr University Bochum
{thorsten.eisenhofer, lea.schoenherr, dorothea.kolossa, thorsten.holz}@rub.de

Abstract

The wide adoption of *Automatic Speech Recognition* (ASR) remarkably enhanced human-machine interaction. Prior research has demonstrated that modern ASR systems are susceptible to *adversarial examples*, i.e., malicious audio inputs that lead to misclassification by the victim's model at *run time*. The research question of whether ASR systems are also vulnerable to *data-poisoning attacks* is still unanswered. In such an attack, a manipulation happens during the *training phase*: an adversary injects malicious inputs into the training set to compromise the neural network's integrity and performance. Prior work in the image domain demonstrated several types of data-poisoning attacks, but these results cannot directly be applied to the audio domain. In this paper, we present the first data-poisoning attack against ASR, called VENOMAVE. We evaluate our attack on an ASR system that detects sequences of digits. When poisoning only 0.17 % of the dataset on average, we achieve an attack success rate of 86.67 %. To demonstrate the practical feasibility of our attack, we also evaluate if the target audio waveform can be played over the air via simulated room transmissions. In this more realistic threat model, VENOMAVE still maintains a success rate up to 73.33 %. We further extend our evaluation to the *Speech Commands* corpus and demonstrate the scalability of VENOMAVE to a larger vocabulary. During a transcription test with human listeners, we verify that more than 85 % of the original text of poisons can be correctly transcribed. We conclude that data-poisoning attacks against ASR represent a real threat, and we are able to perform poisoning for arbitrary target input files while the crafted poison samples remain inconspicuous.

1 Introduction

Digital voice assistants are ubiquitous, whether at our homes, in our cars, or on our smartphones. Forecasts predict that by 2024 the number of digital voice assistants will surpass the world's population with more than 8 billion devices [52]. While there is constant effort to improve the robustness of their built-in *Automatic Speech Recognition* (ASR), voice assistants still show unexpected behavior due to misinterpretations of given audio inputs, e. g., accidental triggers [15, 41] or crafted music [30]. An attacker that actively exploits these blind spots can violate both security and privacy assumptions. In fact, prior research has demonstrated that ASR systems are susceptible to malicious inputs, such as adversarial examples [1, 11, 42]. In these attacks, an audio file is perturbed by imperceptible amounts of noise to trigger a misclassification by the victim's ASR system at *run time*. While adversarial examples are a well-studied phenomenon and have been demonstrated to work for various domains [4, 6, 11, 19, 22, 34, 57], attacks *during training* of an ASR, the so-called *data-poisoning attacks* [8, 18, 33], have not been studied, yet. Such attacks compromise the training data of a model with the intention of misclassifying specific inputs after the deployment of the trained model.

Machine learning models, including ASR, require a huge amount of training data and it is common practice to collect these datasets from potentially untrustworthy sources (e. g., through crowd-sourcing or using open-source repositories). State-of-the-art ASR systems require thousands or even millions of samples, therefore, it is infeasible to thoroughly and manually verify a full training set. Although several automated defenses have been proposed [12, 31, 38], these sanitization-based defenses may be overwhelmed by some attacks [27, 43], as the defenses are attack-specific.

Another pitfall is the fact that the training data is in general not released with a model's deployment and—in contrast to adversarial examples—poisoning attacks are hard to detect during run time as the input is unaltered [13, 24]. Especially in privacy-preserving *federated learning* scenarios, where the training data and the training is decentralized, a party can easily compromise the training data [50].

Despite these threats and although a recent survey of 28 industry organizations found that industry practitioners ranked *data poisoning* as the most serious threat to ML systems [28], poisoning attacks are a neglected—yet critical—attack scenario. Especially for speech-based systems, poisoning attacks

have not been investigated, and recent research papers only focus on adversarial examples and their countermeasures [1].

Targeted clean-label poisoning attacks have been proposed against image classification systems [2,17,21,44,58], wherein the poison samples are perturbed to trigger the system's misbehavior for specific target inputs. In these attacks, the adversary has *no* control over the labeling process, and the poison samples have correct labels, therefore, they are inconspicuous. Compared to other types of poisoning attacks [20, 33], which aim to downgrade the baseline test performance, the poisoned model of a targeted attack usually maintains the same level of test performance.

Poisoning ASR systems is significantly more challenging than other domains, such as images: ASR systems map an audio waveform into a *sequence* of words and thus consider a time series as input; as a result, an ASR system works quite differently than an ordinary image classifier. Additionally, ASR systems are—in general—trained from scratch. Previous targeted poisoning attacks against image classification systems have mostly focused on a transfer learning scenario, where a pre-trained and *frozen* network acts as a feature extractor, and an application-specific linear classifier is fine-tuned. For these reasons, attacks from the image domain cannot be naïvely applied to ASR systems.

In this paper, we present the design and implementation of VENOMAVE, the first poisoning attack in the audio domain. More specifically, we study different aspects of this attack and demonstrate that such an attack is feasible against state-of-the-art ASR systems. To recognize each word, ASR systems divide an audio waveform into overlapping frames and process all these frames individually, which results in a sequence of states. Given this, a targeted misclassification of any utterance into a word can be interpreted as a series of misclassification of these states. Leveraging the time-series properties of audio signals, we divide our poisoning goal into a series of misclassification tasks; in each of them, we focus on poisoning one state.

In particular, VENOMAVE can be divided into three steps: First, we select a target sequence of states that an ASR system typically produces for an utterance of our target word. To choose such a sequence, we iterate over the training samples that contain the word and record the frequency of states. Afterwards, the poison samples are selected by utilizing a poising budget to divide them across the misclassification tasks. Finally, the poison samples are crafted via *surrogate models* that are trained from scratch with different random seeds at each step of the poison optimization, such that their malicious characteristics will transfer to the victim's model, which will be trained from scratch on the poisoned dataset.

In our experiments, we target a specific utterance to perform a *word replacement attack* on the TIDIGITS dataset [29], a dataset of uttered digit sequences of different length. We craft poison samples for a specific utterance of a digit (e. g., `EIGHT` such that the system recognizes it as `NINE`). To determine whether the attack is successful or not, we train the victim's system on the poisoned dataset and evaluate it against the target utterance. Additionally, we evaluate the performance of the poisoning attacks for an adversary that has no information about the victim's network architecture and parameters as well as the victim's training set. We show that VENOMAVE can successfully craft poison samples that lead to a targeted misclassification if they are used during the victim's training. In further experiments, we demonstrate that the attack works in over-the-air scenarios by simulating a potential transmission in a room. We also utilize psychoacoustic hearing thresholds [42, 59] to limit the perceptible perturbations and therefore the inconspicuousness of the poisoned data. To examine the practical feasibility and scalability of our attack, we successfully apply VENOMAVE against the larger *Speech Commands* dataset [53]. The results show that the attack is also successful in this scenario.

Finally, we conducted a user study in which we asked human participants to transcribe the poisoned data generated by VENOMAVE. On average, more than 85 % of the poison samples were transcribed as their original labels, which shows that VENOMAVE is able to generate clean-label poison samples. It is worth noting that such a study is missing in the evaluation of clean-label poisoning attacks in the image domain. As a recent benchmark acknowledges [43], current attacks, which claim to generate inconspicuous poison samples, often produce easily visible image artifacts and distortions.

In summary, we make the following key contributions:

- **Poisoning against ASR.** We propose the first targeted poisoning attack in the audio domain and demonstrate that data-poisoning attacks are a real threat for ASR systems. Having poisoned only 25.44 seconds of audio (0.17 % of the training set) on average, VENOMAVE achieves an attack success rate of 86.67 % for the TIDIGITS dataset. For the Speech Commands dataset, on average 116.73 seconds of audio (0.14 % of the training set) is enough to achieve an attack success rate of 80.0 %.

- **End-to-end Attack.** In our threat model, we assume the victim's system is trained on the poisoned data from scratch, which is a much more challenging scenario compared to attacks against transfer learning. We successfully demonstrate a full end-to-end attack.

- **Transcription Test.** We perform a transcription test with human listeners on the crafted poisons. Our results show that human listeners are able to transcribe the original text for more than 85 % of the poison samples.

To support further research in this area, we release the source code of all experiments as well as the poison samples generated by VENOMAVE at https://github.com/usenix-author2022/anonymous-submission.

2 Related Work

In the following, we discuss related work on attacks against machine learning and ASR systems.

Adversarial Examples. Adversarial examples are inputs that have been perturbed by adding imperceptible noise to fool a machine learning classifier [7, 48]. Such perturbations are calculated using the gradients of an optimization problem that is defined on the victim network, or surrogate networks, if the victim network is unknown. Initial work on adversarial attacks focused on the space of images [7, 19, 48]. Later, similar evasion attacks were shown to exist in the speech recognition domain, where generating adversarial examples is more challenging due to time dependencies that exist in the ASR systems [1, 10, 11, 40, 42, 51, 55].

Data-Poisoning Attacks. The first data-poisoning attacks aimed to degrade the test accuracy of the victim model [8, 33, 35, 54]. These attacks are generally easy to detect, as the model can always be assessed by evaluating against a private test set. In *backdoor attacks* [20], an adversary fools the victim model by imprinting a small number of training samples with a specific pattern (*trigger*) and changing their labels to the desired target label. The attacker can achieve misclassification by injecting the trigger into *any* input example.

Clean-label poisoning attacks against image classification systems [2, 17, 21, 44, 58] are the attacks that are closest to our work, wherein the adversary has *no* control over the labeling process. In these attacks, the poison samples are perturbed to achieve the system's misbehavior for specific inputs. These attacks target image classifiers, for which the attack succeeds if a single misclassification happens.

While clean-label poisoning attacks in the image domain have been successfully demonstrated, the recognition process of ASR is based on time series signals (i. e., the waveform audio signal), and, therefore, these poisoning attacks are not applicable to speech-based systems.

3 Technical Background

In the following section, we briefly outline the fundamental concepts of ASR systems and formally introduce data poisoning attacks.

3.1 Speech Recognition Systems

Speech recognition systems can be of two kinds: end-to-end systems and hybrid systems. End-to-end systems refer to architectures where the neural network directly transforms the audio waveform (input) into a character transcription (output). In contrast to an end-to-end system, a hybrid ASR system combines two models, a *Deep Neural Network* (DNN) for acoustic modeling and a *Hidden Markov Model* (HMM) that is used as the language model for cross-temporal information integration, after the input has been initially processed by the DNN. Hybrid systems still represent the state-of-the-art approach in ASR, given that they are not as data-hungry as end-to-end systems [5]. Furthermore, their architecture allows the user to explicitly specify task grammars and adapt language models on the fly. Additionally, hybrid ASR systems still provide the best results in practice and in typical benchmark tasks [14, 25, 32]. Such advantages have made them very popular for many commercial systems such as, Amazon's Alexa, and the system of choice for pre-trained embedded models that may not have the luxury of a big-data backend for adaptation [41].

Figure 1 provides an overview of the main system components. In hybrid ASR systems, DNNs are typically used as an acoustic model to describe the probabilities of HMM states: The HMM mainly describes the language grammar, a phonetic description of all words, and context-dependencies of phonetic units and words.

(i) **Feature Extraction.** The raw waveform input is typically processed into a feature representation that should ideally preserve all relevant information (e. g., phonetic information that describes the smallest acoustic unit of speech) while discarding the unnecessary remainders (e. g., acoustic properties of the room). We divide the input waveform into overlapping frames of fixed length, and process each frame using the *Discrete Fourier Transform* (DFT) to obtain a frequency representation in the form of a spectrogram. We further calculate commonly used *Mel Frequency Cepstral Coefficients* (MFCCs) features [47], which consider the logarithmic frequency perception of the human auditory system. For this purpose, a Mel-scaled filterbank [45] is applied to the spectrogram representation, and the output is further processed by applying a *Discrete Cosine Transformation* (DCT). The resulting MFCC features are augmented by their first and second derivatives to represent the temporal structure of the input.

(ii) **Neural Network.** In hybrid ASR systems, DNNs are typically used as an acoustic model to describe the probabilities of HMM states. The HMM mainly describes the language grammar, a phonetic description of all words, and context-dependencies of phonetic units and words.

(iii) **Decoding.** The decoding in hybrid ASR systems utilizes some form of graph search on top of the neural network's output to infer the most probable word sequence from the acoustic signal. For this purpose, the HMM is used to find an optimal path (which is interpreted as a sequence of words) through the HMM via dynamic programming, such as Viterbi decoding [37] or beam search [36].

Viterbi training. During training of an ASR system, the exact alignment between utterances and transcriptions and

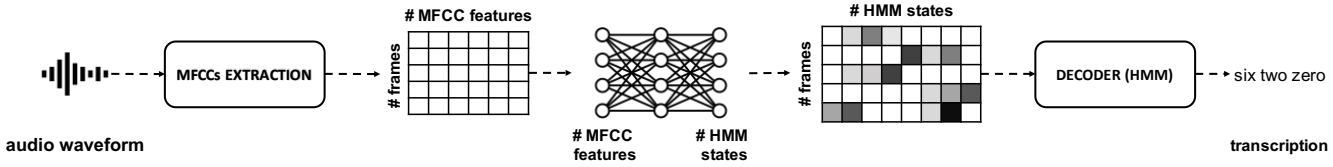


Figure 1: Overview of a state-of-the-art hybrid ASR system with the two main components: the neural network acts as an acoustic model and the decoder employs an HMM to generate the transcription. The HMM mainly describes the language grammar, a phonetic-based word description of all words, and context-dependencies of phonetic units and words.

therefore the labels are typically unknown. To account for this, *Viterbi training* is commonly utilized. Starting with training on equally aligned labels, an initial neural network is trained, followed by a decoding of the training data. The resulting alignment is further used as the new alignment of the utterance.

3.2 Data Poisoning

To craft poison samples, we exploit the mathematical guarantee that any *linear* classifier that associates a set of samples P to class l will also classify any point inside their convex hull as class l. To explain the attack, we divide the network into two parts: (1) all layers up to the penultimate layer, which act as a feature[1] extractor network Φ, and (2) the last layer, which is a linear classifier. The victim's model will identify the target frame x_τ as the target class Z, if the target frame lies within the convex hull of the poison frames $\{x_\gamma^{(p)}\}_{p=1}^P$ with class Z (in the feature space created by the feature extraction model Φ).

Specifically, we solve the following optimization:

$$\min_{\{x_\gamma^{(p)}\}} \frac{1}{2M} \sum_{m=1}^{M} \frac{\left\| \Phi^{(m)}(x_\tau) - \frac{1}{P}\sum_{p=1}^{P} \Phi^{(m)}(x_\gamma^{(p)}) \right\|^2}{\left\| \Phi^{(m)}(x_\tau) \right\|^2} \quad (1)$$

By solving this optimization for similar models it has been shown that such a guarantee will ideally also transfer to the unknown victim network [2, 58]. To solve the non-convex problem in Equation 1 (i. e., find the optimal set of poison samples), we iteratively apply gradient descent to optimize the poison frames $\{x_\gamma^{(p)}\}_{p=1}^P$.

4 Method

In the following, we introduce VENOMAVE, the first data-poisoning attack against ASR systems. For this purpose, we divide the attack into several parts to select the optimal target sequence and their respective poison samples. Generally speaking, we need to address the following two challenges associated with attacking ASR systems:

(i) **Time series.** The input of an ASR system is a time series of samples from the audio signal rather than a single, fixed-sized input. Consequently, also the system's output is a sequence of classes with time dependencies that need to be considered during the attack.

(ii) **End-to-end attack.** ASR systems are typically trained from scratch, whereas a modern image classification system can rely on transfer learning. In terms of the attack, dealing with models that are trained from scratch is a much more difficult task as the attacker needs to take the complete training phase into account.

From a high-level perspective, our goal is to implant manipulated data points into the victim's training set, such that a system that is trained on this data transcribes phrases into attacker-chosen commands. In the following, we first describe our threat model and then present the necessary steps and ideas of VENOMAVE.

4.1 Threat Model

The attacker aims to poison the victim's model to trigger a *targeted* misclassification of a specific utterance as an attacker-chosen sequence of words (e. g., an utterance of `EIGHT` to be recognized as `NINE`). We assume the victim downloads the training set from an untrusted source (e.g., through crowdsourcing or using open-source repositories), which the attacker is able to manipulate. Thus, the adversary can inject poisoned data into the victim's training set.

We evaluate VENOMAVE on hybrid ASR systems that are used throughout the research community as well as in commercial products, such as e. g., Amazon's Alexa [41]. In our threat model, we assume that the victim uses the same training parameters for building the language model; however, the training parameters and architecture of the neural network is unknown to the attacker. We also evaluate VENOMAVE in a weaker threat model, where the adversary does not know the victim's training set, except for the poisoned data.

In our evaluation, it is always assumed that the victim trains the whole ASR system from scratch with an unknown random seed.

[1] Throughout the paper, by the term features, we refer to the features represented by the penultimate layer, not MFCCs (i.e., the acoustic features which are passed as the input to the network.)

4.2 VENOMAVE Algorithm

In this work, we focus on changing exactly one word during the attack (e.g., we change one digit from the original transcription to another one). Given $(\mathbf{x}_t, \mathbf{W})$, the goal is then to find a set of poisons $\mathbf{P}$, s.t. the audio input $\mathbf{x}_t$ is transcribed into the target word $\mathbf{W}$ by *any* system trained on the poisoned dataset. In general, the attack can be divided into three steps:

(ii) *Sequence Selection.* First, we determine which frames of $\mathbf{x}_t$ need to be misclassified.

(ii) *Poison Selection.* For each frame, we select a set of poison frames from one or multiple samples.

(iii) *Poison Crafting.* We alter the selected poison frames (the *poisoned data*) to cause the misclassification.

Figure 2 illustrates the individual steps of the attack in an example, where the ASR system is fooled to recognize an audio waveform of `382` as `392`. The ultimate, but implicit, adversarial label is `NINE` and the original label is `EIGHT`. We use this example throughout this section to explain each step in detail. The full attack is described in Algorithm 1.

With VENOMAVE, we inject poisoned data into the victim's training set such that the trained neural network will generate an adversarial output for the target input, which will be decoded by the language model as the targeted word sequence. Therefore, the explicit adversarial label is a sequence of the HMM states. Note that not only one possible sequence of HMM states would lead to a specific transcription. For this reason, we first have to determine which output sequence is a promising candidate as our target.

Sequence Selection. The language model defines a word $\mathbf{W}$ as a sequence of K states $\mathbf{W}=\{w_k\}_{k=1}^{K}$. Assuming that the sequences for the digits `EIGHT` and `NINE` consist of 5 and 3 states, respectively, the two words can be described with HMM states $\mathbf{EIGHT}=\{8_1,8_2,8_3,8_4,8_5\}$ and $\mathbf{NINE}=\{9_1,9_2,9_3\}$. In general, the number of frames of an uttered word is larger than the number of states K of the corresponding language model, e. g., for the word `NINE` uttered across 6 frames, both sequences $[9_1,9_1,9_2,9_2,9_3,9_3]$ and $[9_1,9_1,9_1,9_2,9_2,9_3]$ would be perfectly valid.

Setting the adversarial labels of these six frames to either of these two sequences will lead to two different sets of poison samples; each aims to fool the neural network to generate a different sequence of adversarial HMM states. To increase the chance that the target sequence will be decoded to the word `NINE`, a sequence should be selected that is more probable from the point of view of the HMM. Hence, we look at the appearances of the word `NINE` in the dataset to select the most common pattern as our sequence of adversarial states.

For state w_i, we define the relative frequency

$$R_{w_i}=\frac{\text{freq}(w_i)}{\text{freq}(\mathbf{W})},$$

where $\text{freq}(w_i)$ and $\text{freq}(\mathbf{W})$ are the total number that state w_i and word $\mathbf{W}$ appear in the dataset. In fact, we have found this selection of adversarial states to be more successful than a uniform selection. Therefore, in our running example, the original sequence $[8_1,8_2,8_3,8_4,8_4,8_5]$ should be changed to $[9_1,9_2,9_2,9_2,9_3,9_3]$, as the state 9_2 appears 3 times more often in the training set than the state 9_1. We then divide our attack into $N=6$ smaller poisoning attacks, described by a set $\mathbf{T}=\{x^{(i)}_{<Y_i,Z_i>}\}_{i=1}^{N}$ of frames $x^{(i)}_{<Y_i,Z_i>}$ with an original state Y_i and an adversarial state Z_i. In our example in Figure 2 the poisoning set is described as

$$\mathbf{T}=\left\{x^{(1)}_{<8_1,9_1>},x^{(2)}_{<8_2,9_2>},\ldots,x^{(N)}_{<8_5,9_3>}\right\}.$$

Poison Selection. Given T, we will craft poison frames separately for each of the state pairs. To achieve this for the attack pair $x_{<Y,Z>}$, we select poison frames with label Z from one or more utterances and change them accordingly such that the frame x_τ will be identified by the poisoned system as state Z. This is repeated for all adversarial frames in $\mathbf{T}$. For the attack pair $x_{<Y,Z>}$, we determine the number of poison frames P_Y based on the frequency of the original states Y:

$$P_Y=\left\lceil \text{freq}(w{=}Y)\cdot r_p \right\rceil,$$

where $0<r_p<1$ describes the *poison budget*. Thus, if an original state Y_i occurs twice as often in the training set as another original state Y_j, we also select twice as many poison frames for the attack $x^{(i)}_{<Y_i,Z_i>}$ than for the attack $x^{(j)}_{<Y_j,Z_j>}$.

The intuition behind this choice of P_Y is that the attack might fail if the target frame x_τ has adjacent neighbor frames from its class Y in the victim's training set. This has also been observed in other poisoning attacks [44, 58]. The poison frames—no matter how well they are crafted—need to compete with these neighbor frames in order to inject the malicious decision boundaries during the training phase.

Poison Crafting. For the attack pair $x_{<Y,Z>}$, the victim's model will identify the target frame x_τ as state Z, if the target frame lies within the convex hull of the poison frames x_γ (in the feature space created by the feature extraction model Φ).

Targeted poisoning attacks in the image domain [2, 58] mainly exploited linear transfer learning, where a pre-trained and *frozen* network acts as a feature extractor Φ, and an application-specific linear classifier is fine-tuned. However, these attacks are not directly applicable for targeting ASR systems, as they are—in general—trained from scratch; meaning that the feature space defined by the penultimate layer Φ is also altered during training. We thus follow a different approach to enforce the guarantee given by Equation (1) for M surrogate models (i.e., models trained with the same parameter but different seeds).

To increase transferability, VENOMAVE optimizes poison frames such that not only the target frame is inside the convex

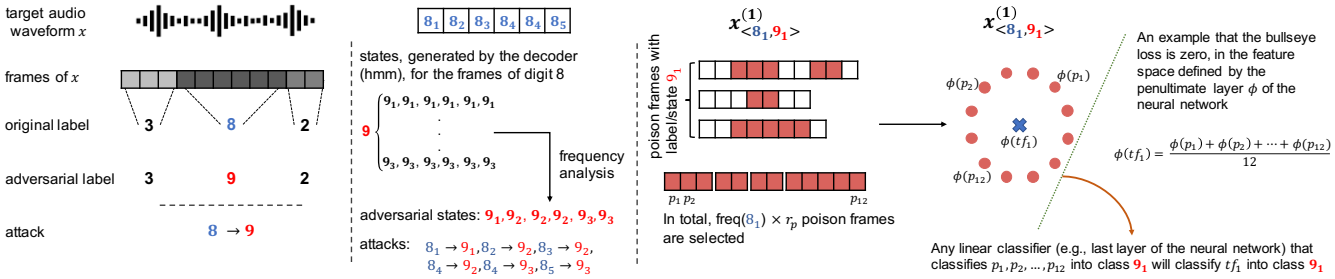


Figure 2: An example of the VENOMAVE attack, where the adversary fools the victim model in producing the transcription `392` for an utterance with the original transcription `382`. The adversary builds a set of surrogate systems and uses them to craft the poison samples, such that their malicious characteristics will transfer to the victim's ASR system.

hull of poison frames, but it is ideally at, or at least close to, the center of this *attack zone*. To cope with this challenge, we train a set of surrogate networks *from scratch* on the current (poisoned) dataset at the beginning of each round of the attack. Subsequently, we modify the poison samples to achieve our desired heuristics with respect to the refreshed surrogate models. Our intuition is that after several rounds of the attack, we reach a state in which the poisoned data need no further modifications to obtain the heuristics, hoping that their malicious characteristics will transfer to the victim's poisoned model.

5 Evaluation

We implemented the proposed attack based on the approach described in Section 4. In this section, we empirically verify and assess the efficacy of VENOMAVE with several experiments. We first evaluate VENOMAVE on an ASR system that recognizes sequences of digits using the TIDIGITS dataset [29]. In our experiments, we target a specific utterance to perform a *word replacement attack*. For example, we craft poison samples for a specific utterance of a digit, e. g., `EIGHT`, such that the system recognizes it as a different one, e. g., `NINE`. Additionally, we evaluate the performance of the poisoning attacks when the adversary has limited knowledge about the victim's network architecture, training parameters, and training set. In further experiments, we utilize psychoacoustic hearing thresholds [42, 59] to limit the perceptible perturbations and therefore the conspicuousness of the crafted poisoned data. We demonstrate that the attack also works in over-the-air scenarios by simulating a potential transmission in different room settings. We further evaluate the practical feasibility of VENOMAVE on a larger ASR system that is trained on the Speech Commands dataset [53].

Before discussing the results, we first describe the standard measures that we use to assess the quality of the poison samples, both in terms of efficiency as well as conspicuousness.

Algorithm 1 VENOMAVE

Input:

$\mathbf{x}_t$ ▷ Target audio waveform
$\mathbf{W}$ ▷ Adversarial word sequence
Ψ ▷ The training dataset
H ▷ Surrogate HMM

$\mathbf{T} \leftarrow$ get_adv_states_pairs($\mathbf{x}$, $\mathbf{W}$) ▷ Select the N pairs of frames that needs to be changed
$[x_\gamma]^N \leftarrow$ select_poisons($\mathbf{T}$) ▷ For each pair of <Y, Z>, we select P_Y poison frames x_γ
for $i = 1$ **to** Q **do**
  for $m = 1$ **to** M **do**
    $\Phi^{(m)} \leftarrow$ train_model(Ψ)
  end for
  for $j = 1$ **to** R **do**
    $\mathcal{L}^{(j)} \leftarrow 0$ ▷ Set loss to zero
    for x^p_γ **in** $[x_\gamma]^N$ **do**
      $\mathcal{L}^{(j)} \leftarrow \mathcal{L}^{(j)} +$ Eq1_loss($x^p_\gamma, x^p_b, x_\tau, [\Phi]^M$) ▷ Equation (1)
    end for
    $\mathcal{L}^{(j)} \leftarrow \frac{\mathcal{L}^{(j)}}{\text{len}([x_\gamma]^N)}$ ▷ Average of the loss
    $[x_\gamma]^N \leftarrow$ update($[x_\gamma]^N, \nabla\mathcal{L}^{(j)}$) ▷ update all poison frames $[x_\gamma]^N$ using $\nabla\mathcal{L}^{(j)}$
    break if $\mathcal{L}^{(j-1)} - \mathcal{L}^{(j)} < 0.0001$
  end for
  $\Psi \leftarrow$ updated(Ψ, $[x_\gamma]^N$) ▷ update Ψ with new poisons $[x_\gamma]^N$
  $\Phi_V \leftarrow$ train_model(Ψ)
  $\hat{\mathbf{W}} \leftarrow$ eval_victim(H, $\mathbf{x}_t$, H) ▷ get word sequence recognized by Φ_V
  break if $\hat{\mathbf{W}} = \mathbf{W}$
end for

5.1 Metrics

Attack Success Rate. In all experiments, VENOMAVE aims to induce a targeted misclassification for a single utterance, when the victim's model is trained on the poisoned dataset. If the targeted misclassification is not triggered, we consider the attack as failed. Therefore, the *attack success rate* describes the percentage of successful attacks.

Clean Test Accuracy. We evaluate the victim's performance against the standard test set to calculate the *clean test accuracy* of the model. An ideal poisoning attack does not degrade the model performance for non-target inputs; otherwise, it might be suspicious. For all test samples, given the model transcriptions, we count and accumulate all substituted words S, inserted words I, and deleted words D over the entire test set, to calculate the accuracy via

$$\frac{N-I-S-D}{N}, \tag{2}$$

where N is the total number of words in the test set's ground-truth labels.

Segmental Signal-to-Noise Ratio (SNRseg). This metric measures the amount of noise σ added by an attacker to the original signal $\mathbf{x}$ and is computed via

$$\text{SNRseg(dB)} = \frac{10}{K}\sum_{k=0}^{K-1}\log_{10}\frac{\sum_{t=Tk}^{Tk+T-1}\mathbf{x}^2(t)}{\sum_{t=Tk}^{Tk+T-1}\sigma^2(t)}, \tag{3}$$

where T is the segment length and K the number of segments. Thus, the higher the SNRseg, the *less* noise has been added. We use a frame length of 12.5 ms, which corresponds to $T = 200$ at a sampling frequency of 16 kHz. As it is computed frame-wise, the SNRseg gives a better assessment of an audio signal than the signal-to-noise ratio (SNR), if the original signal and the modified signal are aligned [56], which is the case in our experiments. As only a very small part of the poison files are changed, we measure the SNRseg only for the poisoned frame.

Maximum Perturbation. We compute the maximum amount of perturbations added by the attack as follows:

$$\delta_{max} = \max\left|\frac{\mathbf{x}}{\nu} - \frac{\mathbf{y}}{\nu}\right|, \tag{4}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the original and poisoned audio waveforms, respectively, and ν is the maximum absolute value of all samples in the training dataset. Therefore, the *maximum perturbation* describes the relative maximum difference between the original and the poisoned file.

5.2 Experiments – TIDIGITS

Dataset Description. The TIDIGITS dataset [29] is designed for speaker-independent recognition of digit sequences. The dataset includes 8,623 and 4,390 utterances in its training and test set, respectively. It is recorded in a quiet environment with a sample rate of 16 kHz. The dataset considers eleven words: `ONE`, `TWO`, ..., `NINE`, `ZERO`, and `OH`. The sequences are spoken by 225 speakers (111 men and 114 women) which are split equally into disjoint sets such that no utterance of a speaker used in the training set appears in the test set. Table 1 presents the speaker statistics of the dataset.

Table 1: TIDIGITS statistics. The number of male and female speakers in the training and test sets of the TIDIGITS corpus.

	# Speakers	
Set	**Man**	**Woman**
Train	55	57
Test	56	57

5.2.1 Experimental Setup

For the experiments, we randomly sample 30 single-digit audio files and a target digit for each. In general, the random seed used by the victim for model training is unknown, and, thus, the parameters of the victim's ASR system, the neural network and the HMM—which depends on the neural network due to Viterbi training—are different from the parameters of the surrogate models.

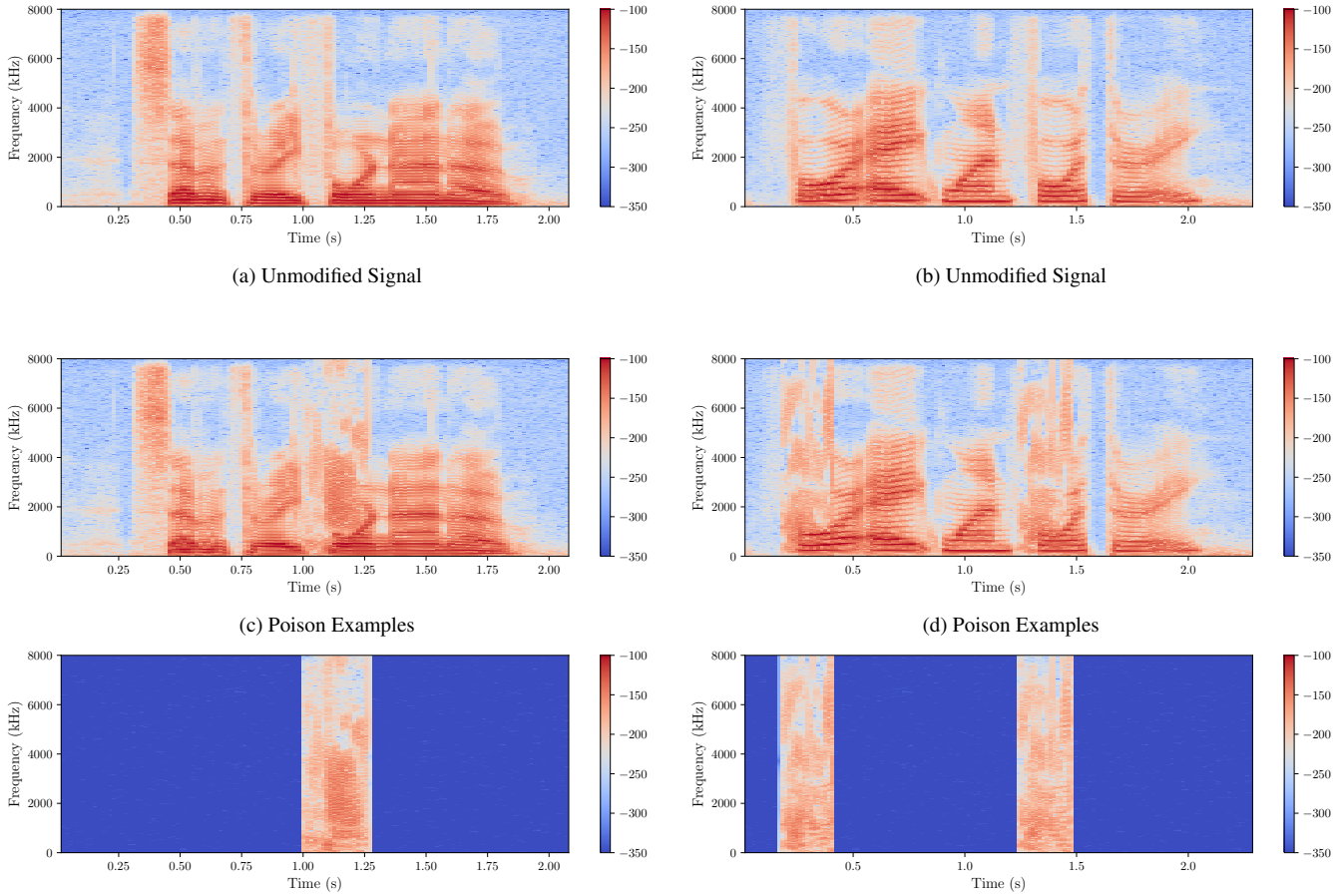
Unless explicitly stated otherwise, we use the following settings across all experiments: We use the DNN_{2+} architecture for our surrogate networks, as described in Table 2. Additionally, we freeze the HMM during the attack to accelerate the attack, as the language model does not typically change significantly. This HMM is created in advance by training an ASR system for 15 epochs on the clean training set, followed by three epochs of Viterbi training. Using this surrogate HMM, at the beginning of each step of the attack, VENOMAVE uses an Adam [26] optimizer with a learning rate of 0.0001 to train M surrogate networks on the latest version of the poisoned dataset for 25 epochs with the batch size of 32.[2]

5.2.2 Experimental Results

Surrogate Models. Table 3 presents the performance of VENOMAVE when it uses different numbers of surrogate networks. We used a full-knowledge threat model, where the victim trains the ASR system from scratch with the same settings that we use to build the surrogate networks. The victim's ASR system is trained for 33 epochs,[3] of which three epochs include Viterbi training. In our analysis, we found using eight surrogate networks has the highest attack success rate (86.67 %).

[2] Training converges at 25 epochs.

[3] Selected to maximize the clean test accuracy.



(a) Unmodified Signal

(b) Unmodified Signal

(c) Poison Examples

(d) Poison Examples

(e) Difference

(f) Difference

Figure 3: **Spectrograms of poison examples.** Figure 3a and 3b show two examples of unmodified signals, Figure 3c and 3d depict the poison version of the two examples, Figure 3e and 3f show the respective differences of both versions.

Given that increasing M adds to the complexity of Equation 1 and the training time and that $M = 8$ surrogate networks performed the best, we choose eight surrogate networks across all experiments unless explicitly mentioned otherwise.

Poison Budget r_p vs. Attack Success Rate. We also evaluate VENOMAVE for varying levels of poison budget r_p (see Section 4.2). Table 4 shows that successively increasing the poison budget from 0.001 to 0.01, and, hence, the number of poison frames, leads to a higher attack success rate (23.33 % $\rightarrow$ 83.33 %). But this improvement comes at a price; the length of the poisoned data increases (6.20 s $\rightarrow$ 48.73 s).[4] Nevertheless, with a budget $r_p = 0.005$, we poison only 0.17 % of the training set while achieving an attack success rate of 86.67 %.

We also measure the SNRseg for the poison samples, as well as the maximum perturbation added to a frame in Table 4. Note that the SNRseg is only computed for the poisoned frames (i.e., clean parts of the poison samples are explicitly excluded) to provide a fair assessment of the added noise. As we only poison 100-200 ms of each poison sample, the overall quality of the poison is much higher.

Figure 3 shows two examples of poisoned audio files. We show the original audio files (Figure 3a and 3b) and the respective poisoned versions (Figure 3c and 3d) in the frequency domain. The bottom pane shows the respective difference of the two versions, (Figure 3e and 3f). The left audio file contains the digit sequence `SEVEN, THREE, FOUR, NINE, OH` and poisons the digit `FOUR` to `TWO`. The one on the right contains the sequence `FOUR, EIGHT, ONE, FOUR, THREE` and poisons the digit `FOUR` to `OH`.

Psychoacoustic Margin Λ vs. Attack Success Rate. Within the same threat model, we utilize psychoacoustic modeling in this experiment to assess its effects on VENOMAVE. Recent adversarial examples against ASR systems [39, 42] practiced psychoacoustic hiding to create less perceptible adversarial noise. To identify inaudible ranges, these attacks use dynamic hearing thresholds, which describe the masking effects in human perception that arise as a function of the interactions between different co-occurring acoustic frequencies. To limit

[4] The total length of the training data is 15,254 s.

Table 2: Neural network architectures used for the TIDIGITS dataset. Networks use two or three hidden layers, each with a softmax output layer of size 95, corresponding to the number of HMM states. The baseline test accuracy is for when the victim uses a clean dataset.

Name	Description	# Parameters	Baseline test accuracy (%)
DNN_2	Two hidden layers, each with 100 neurons	54,895	98.75
DNN_{2+}	Two hidden layers, with 100 and 200 neurons	100,095	98.79
DNN_3	Three hidden layers, each with 100 neurons	64,995	98.41

Table 3: Evaluation of VENOMAVE when it uses different numbers of surrogate networks. The psychoacoustic modeling is disabled, and r_p is set to 0.005. This experiment has been done using `NVIDIA RTX A6000` graphics cards (with `CUDA 11.0`, `PyTorch 1.9.1`, and `Torchaudio 0.9.1`). When training the surrogate networks, each graphics card is used to train up to four networks. We found this selection more efficient in comparison to training one network at a time.

	M					
	1	2	4	6	8	10
# Attack step (Q)	15.73	11.47	7.9	7.6	6.83	7.0
Attack time (in hours)	1.54	1.36	1.46	3.43	3.33	5.33
Attack success rate (%)	43.33	76.67	80.00	80.00	86.67	83.33
Clean test accuracy (%)	97.84	97.84	97.81	97.79	97.84	97.81

Table 4: Evaluation of VENOMAVE when the poison budget r_p is successively increased from 0.001 to 0.01. Increasing the number of poison frames leads to a higher attack success rate. However, improvement comes at a price; the length of the poisoned data increases. The total length of the training data is 15.254 s.

	r_p			
	0.001	0.003	0.005	0.01
Poisoned frames SNRseg	3.06	2.38	2.17	3.76
Maximum perturbation δ_{max}	0.16	0.16	0.17	0.16
Poisoned data length (in seconds)	6.20	15.93	25.44	48.73
# Poisoned data samples	96.23	248.10	387.83	693.57
Attack success rate (%)	23.33	76.67	86.67	83.33
Clean test accuracy (%)	97.85	97.84	97.84	97.76

audible distortions, we implement psychoacoustic hiding similar to what is described by Schönherr et al. [42]. Appendix A elaborates in detail how we limit our attack by psychoacoustic filtering.

In particular, we evaluate VENOMAVE for varying degrees of psychoacoustic filtering Λ, which is controlled through the margin (in dB) that we allow the attack to surpass the hearing thresholds. The higher Λ, the more audible noise is allowed to be added by the attack. It should be noted that the choice of poison samples and frames does not depend on the margin Λ.

As shown by Table 5 for $r_p = 0.005$, enabling the psychoacoustic hiding with a margin $\Lambda = 30$ dB decreases the attack success rate from 86.67 % to 43.33 %, while the SNRseg of poisoned frames improves (2.17 dB $\rightarrow$ 4.25 dB). The case without enforcing hearing thresholds is denoted in the table as `NONE`.

In general, with decreasing Λ the attack success rates drop, while the generated poison frames have slightly larger SNRseg values with respect to the original frames. Although enabling psychoacoustic hiding leads to higher SNRseg values, and therefore, less noise, the trend is not as distinct as what is observed by Schönherr et al. [42]. Section 5.4 further elaborates on the human perception of our poison samples based on a user study.

Limited-Knowledge Adversary. So far, we have discussed the results for a full-knowledge adversary, who has complete knowledge of the training parameters used by the victim for both, the neural network and the language model. In this section, we simulate an attacker with limited knowledge. For our evaluation, we use the poisoned data from previous experiments with $r_p = 0.005$ and no psychoacoustic hiding.

In this more realistic threat model, the victim uses different training parameters as well as different DNN architectures. We assume that the victim uses either DNN_2 or DNN_3 (see Table 2). Both of these architectures have fewer parameters in comparison to DNN_{2+}, which is used by the attacker.

For training, the victim uses a learning rate of 0.0004, a batch size of 64, and a dropout probability of 0.2[5] to train the ASR system from scratch for 32 epochs, of which two epochs include Viterbi training.

Dropout randomization is a technique to prevent neural networks from overfitting and increase their performance [46]. It should be noted that in previous clean-label poisoning works, dropout was typically disabled, as in a transfer learning scenario, a rational victim will usually overfit the training set [2, 44, 58]. Since this is often not the case when the victim's model is trained from scratch, we enable dropout in this experiment. With the settings that are used by the attacker for surrogate networks, the victim achieves a test accuracy of 98.75 % and 98.41 % for DNN_2 and DNN_3, respectively, while the baseline test accuracy is 98.69 % and 98.77 % for DNN_2 and DNN_3, respectively.

Table 6 shows that the malicious characteristics of the poisoned data crafted by VENOMAVE will remain even if the victim uses different training parameters and network architectures.

[5] The dropout layer is added after the first hidden layer.

Table 5: Evaluation of VENOMAVE when different levels of poison budget r_p and psychoacoustic filtering Λ are used. When r_p is fixed, the same poison samples and frames are used regardless of Λ. In this experiment, the victim ASR system is built from scratch with the same settings that were used to build the surrogate networks, however, with a different random seed.

r_p	Poisoned data length (seconds)	Poisoned data samples	Λ (dB)	Poisoned frames SNRseg	Max perturbation δ_{max}	Attack success rate (%)	Clean test acc. (%)
0.01	48.73	693.57	10	7.59	0.15	0.00	97.76
			20	5.32	0.21	10.00	97.70
			30	5.09	0.20	60.00	97.74
			40	4.91	0.16	73.33	97.73
			50	4.98	0.15	90.00	97.76
			NONE	3.76	0.16	83.33	97.76
0.005	25.44	387.83	20	4.61	0.17	0.00	97.80
			30	4.25	0.18	43.33	97.80
			40	3.54	0.17	66.67	97.81
			50	4.13	0.15	80.00	97.80
			NONE	2.17	0.17	86.67	97.84

Table 6: Limited-knowledge adversary: Evaluation of VENOMAVE when the adversary has limited knowledge of the victim's training setting. The poison budget r_p is set to 0.005, while psychoacoustic filtering is disabled . For comparison, the first row represents the full-knowledge threat model. The second and third rows present the performance of the attack, when the victim uses a different network architecture, DNN_2 and DNN_3, respectively. Rows 4-6 show the attack statistics in a weaker threat model, in which the adversary does not know the training parameters and the neural network architecture.

Attacker settings	Victim settings: Network	Learning rate	Dropout	Batch size	Epochs	Attack succ. rate (%)	Clean test acc. (%)
Network=DNN_{2+}, batch size=32, dropout=0.2, learning rate=1e-4, DNN training epochs=25N	DNN_{2+}	1e-4	-	32	15N+3V+15N	86.67	97.84
	DNN_2	1e-4	-	32	15N+3V+15N	80.00	97.49
	DNN_3	1e-4	-	32	15N+3V+15N	83.33	97.55
	DNN_{2+}	4e-4	0.2	64	10N+2V+20N	83.33	97.94
	DNN_2	4e-4	0.2	64	10N+2V+20N	86.67	97.92
	DNN_3	4e-4	0.2	64	10N+2V+20N	86.67	98.04

Table 7: Evaluation of VENOMAVE when the adversary has no or partial knowledge of the victim's clean training samples. In this experiment, we assume the victim uses different training parameters (similar to Table 6) compared to what are used by the attacker. We divide the training set of TIDIGITS into two subsets, with "Split 1" containing the first half and "Split 2" the second half of the speakers (56 speakers each).

Attacker settings: Network	Attacker settings: Training set	Victim settings: Network	Victim settings: Training set	Attack success rate (%)	Clean test accuracy (%)
DNN_{2+}	Split 1	DNN_3	Split 2	86.67	97.92
		DNN_3	Split 1 + 2	80.00	98.03

Limited Knowledge of Victim Training Set. So far, we assumed that the adversary knows the victim's training data. In this section, we extend our analysis of VENOMAVE by evaluating it for a more realistic threat model, where the adversary does not know the complete training set of the victim, except for the injected poisoned data.

For the experiments in Table 7, we restrict the adversary to use only 50 % of the speakers of the TIDIGITS training set (Split 1, 56 speakers). Similar to the previous experiment, we evaluate a victim with different training parameters and network architecture (DNN_3) compared to the attacker settings. However, this time, we used two different training sets for the training of the victim's network; (2) training samples from the remaining 56 speakers (Split 2), and (1) the entire training set (Split 1+2). For both cases we have used the same poisoned data.

Table 7 presents the performance of VENOMAVE for these two scenarios. When the victim's training set has no overlap with the attacker's training set, VENOMAVE achieves an attack success rate of 86.67 %. When the attacker's training set consists of 50 % of the victim's training set (second row in Table 7) VENOMAVE achieves an attack success rate of 80.0 %. It should be noted that the same poisoned data is used in these two cases, but in the latter, the poisoned data are competing with more clean data points. This might explain why VENOMAVE achieves a lower attack success rate despite the fact that it has a partial knowledge of the victim's training set.

5.3 Experiments – Speech Commands

To further examine the practical feasibility of our attack, we evaluate VENOMAVE on a larger ASR system. To this end, we use the Speech Commands corpus [53], which is mainly built for the task of keyword spotting.

Dataset Description. The Speech Commands dataset consists of 105,829 one-word utterances, recorded with a sample rate of 16 kHz. The dataset contains 35 different words, which are (1) digits `ZERO`, ..., `NINE`, (2) ten common words for IoT or robotics applications (`YES`, `NO`, `UP`, `DOWN`, `LEFT`, `RIGHT`, `ON`, `OFF`, `STOP`, and `GO`), and (3) four command words (`FORWARD`, `FOLLOW`, `BACKWARD`, and `LEARN`). There are also eleven auxiliary words added to the dataset (`BED`, `BIRD`, `CAT`, `DOG`, `HAPPY`, `HOUSE`, `MARVIN`, `SHEILA`, `TREE`, `VISUAL`, and `WOW`).

To cope up with the larger dataset, we use a larger neural network as well as a larger language model. In particular, we utilize an HMM with 350 states.

Surrogate Models. We use the DNN_{3+} architecture for our surrogate networks, which consists of three hidden layers with 400, 300, and 200 neurons. Similar to the TIDIGITS experiments, we use a fixed HMM during the attack, which is created in advance by training an ASR system for 16 epochs on the clean training set, of which the last epoch enables Viterbi training. We use this surrogate HMM at the beginning of each step of the attack to train four surrogate networks on the latest version of the poisoned dataset for 20 epochs with a batch size of 32.[6] Similar to the TIDIGITS experiments, we use the Adam [26] optimizer with a learning rate of 0.0001 for poison crafting.

Victim Model. In our evaluation, the victim uses a network architecture consisting of four hidden layers with 300, 200, 200, and 200 neurons, respectively. The victim trains the ASR system from scratch for 31 epochs, of which the eleventh epoch enables Viterbi training. For the victim's training, a learning rate of 0.0004 and a batch size of 64 is used.

Results. We randomly select 15 audio files and for each sample, we pick a random adversarial target word. We run VENOMAVE with a poison budget of $r_p = 0.02$. Table 8 shows the attack performance for each individual example.

5.4 User Study – Transcription Test

To evaluate the human perception of our poison samples, we have conducted a user study. More specifically, to assess the quality of the poison samples, we asked participants to transcribe utterances of the poisoned data. For the study, we randomly selected 20 poison samples from 12 attack examples that were successful, both when the psychoacoustic hiding was disabled and for $\Lambda = 30$ dB, which resulted in a pool of 480 poison samples. We use $r_p = 0.005$ as our poison budget for these attacks. In addition, we asked each user to transcribe a few clean samples such that we can verify if the user has provided trustworthy assessments.

We asked 23 English speaking persons to transcribe a random subset of the utterances. The participants were not informed if a sample has been modified or if it represent a clean sample. On average, each user transcribed 40 poison samples. After the study, we de-briefed each participant.

For each attack example, we report the ratio of the poison samples that are transcribed into their original label. When the psychoacoustic hiding is disabled, across the 12 successful attacks, 87.08 % of the poison samples were transcribed into their original labels. On the other hand, for $\Lambda = 30$ dB, 85.00 % of the poison samples were transcribed into their original labels. These results show that despite the fact that enforcing hearing thresholds of $\Lambda = 30$ dB might improve the SNRseg values of the poisoned frames (from 2.17 to 4.25, see Table 5), it does not help in transcribing the samples.

The human study results also indicate that the poisoned data generated by VENOMAVE contain samples that can not necessarily be considered as clean-label samples. We believe poisoning attacks need to do a similar assessment before claiming to be "clean-label." Unfortunately, such a study is missing in the evaluation of all clean-label poisoning attacks in the image domain. In fact, as acknowledged by a recent

[6] Training converges at 20 epochs.

Table 8: Evaluation of VENOMAVE on the Speech Commands dataset. This experiment is done for 15 different randomly selected attack examples. The poison budget r_p is 0.02, and the attacker uses four surrogate networks to craft the poisoned data. On average, VENOMAVE uses 116.73 seconds of poisoned data (0.14 % of the training set). The total length of the training data is 84,054 seconds. The average SNRseg for poison frames is 4.14.

Original word	Adversarial word	Poisoned data length (seconds)	Poisoned data # samples	Poisoned frames SNRseg	Attack successful?	Clean test accuracy (%)
learn	on	31.59	396	7.99	✓	86.83
nine	four	156.71	1,887	7.49	✓	87.07
three	six	124.71	1,654	-1.74	✗	87.16
six	off	91.55	1,057	-0.63	✓	86.98
yes	go	140.74	1,493	7.75	✓	86.90
six	five	128.36	1,584	7.39	✓	87.72
follow	three	51.06	865	1.72	✗	87.39
four	zero	164.14	2,012	8.37	✓	86.99
follow	two	45.35	549	3.74	✓	86.79
four	yes	184.95	2,153	4.06	✓	87.35
six	seven	217.60	2,412	4.07	✗	87.35
one	forward	80.66	1,064	5.09	✓	85.86
four	up	150.78	1,659	-1.67	✓	86.77
up	off	79.65	1,025	3.07	✗	86.67
one	down	94.10	1,256	5.33	✓	87.12

benchmark [43], these attacks, which claim to generate "clean-label," poisoned data often produce easily visible image artifacts and distortions.

5.5 Over-the-Air Attack

In this section, we simulate potential acoustic sound transmissions of audio signals in order to study their effects on our poisoning attack. Some adversarial attacks [40, 55] consider such transmissions while creating adversarial examples. During such a transmission, the audio signal is altered and, therefore, the adversarial perturbations.

We conduct an experiment to compare the attack success rate for varying room settings, specifically, different reverberation times, room dimensions, microphone positions, and speaker positions. In particular, we have evaluated the attack against three (simulated) different rooms with the microphone and the speaker being positioned randomly. We tested each room for reverberation times 0.4, 0.6, 0.8, and 1.0. We simulate the transmission in a room via a convolution with a *Room Impulse Response* (RIR) [3]. For this, we sample a RIR for a specific room setting using the *Python RIR Simulator* implementation [9].

Table 9 shows the attack success rate when the attacker uses the surrogate network DNN_{2+} and the victim uses DNN_3. Although we do not consider acoustic properties of the room during the optimization of poison samples, VENOMAVE still maintains a success rate of 33.33-73.33% across different room settings. It should be noted that the attack success rate for the original audio file is 86.67% (the first row of Table 7).

6 Discussion

In our experiments, we have shown the general feasibility of data-poisoning attacks against ASR systems that are trained from scratch. This is a more realistic but also more challenging assumption. Nevertheless, during our experiments, we proved that training the victim's model from scratch (and not relying on transfer learning) will not prevent our attack from fooling the victim's model. In the following, we discuss our major findings, limitations of VENOMAVE, and possible defenses against poisoning attacks.

6.1 Major Findings

Attack Parameters r_p and M. In general, using larger values of poison budget r_p would increase the number of poisoned files (and frames). However, we showed that beyond a poison budget of 0.005, the attack success does not further improve (see Table 4), and, therefore, more poison samples are not necessarily of value for the attack. The same can be observed for the number of surrogate models; using more surrogate models does not necessarily increase the attack's success (see Table 3), but adds to the attack complexity.

Over-the-Air. In Section 5.5, we demonstrated that VENOMAVE is also successful after the targeted audio signal is played over the air in different (simulated) room settings. This shows the general robustness of our attack and that the poison samples also remain effective after some modifications.

Table 9: Simulated over-the-air evaluation of VENOMAVE. For each of the 30 attack examples (TIDIGITS), we simulate that the target audio file is played in three different rooms. We selected these room dimensions from related work [49], which proposed a dataset of real room impulse responses. We selected the speaker and microphone positions randomly for each room. We evaluate the attack in each simulated room with reverberation times 0.4, 0.6, 0.8 and 1. Note that the attack success rate for the original audio file is 86.67% (the first row of Table 7).

Room		Microphone	Speaker	Attack Success Rate (%)			
Dimensions (m^3)	Volume (m^3)	Position	Position	RT=0.4	RT=0.6	RT=0.8	RT=1
$10.7 \times 6.9 \times 2.6$	192	$1.0 \times 4.5 \times 1.3$	$8.1 \times 3.3 \times 1.4$	53.33	46.67	36.67	33.33
$4.6 \times 6.9 \times 3.1$	98	$3.8 \times 3.2 \times 1.2$	$3.8 \times 5.3 \times 1.0$	63.33	60.00	50.00	46.67
$7.5 \times 4.6 \times 3.1$	107	$0.4 \times 0.9 \times 1.1$	$6.9 \times 1.9 \times 2.6$	73.33	60.00	56.67	56.67

Larger Vocabulary. Our experiments with the Speech Commands dataset showed that VENOMAVE scales to ASR systems with a larger language model, too. This result shows the practical feasibility of our attack.

6.2 Limitation – Psychoacoustic Hiding

Although hearing thresholds have shown to be effective for adversarial examples, in the case of poisoning, their effect is less distinct. One main reason may be that in contrast to adversarial examples, where the complete file is modified, modifications of a poison file are limited to short sequences.

The poison samples calculated with VENOMAVE are not always clean-label data points, even though our transcription study shows that more than 85.0 % of the generated poisoned data have clean labels. It is worth noting that such a study is missing in the evaluation of clean-label poisoning attacks in the image domain. Additionally, a recent benchmark acknowledges [43], that these attacks often produce easily visible image artifacts and distortions.

Nevertheless, in privacy-preserving *federated learning* scenarios, where the training data and the training is decentralized, a party can easily compromise the training data [50]. Here, the poison samples are not constrained to clean-label data points, as the victim has no access to the training data, while the attacker has full control of their data. Additionally, our limited-knowledge experiments have shown that controlling only parts of the training process and training data—as would be the case in a federated learning scenario—is very effective.

6.3 Defenses

Related work published defense techniques that detect poison samples and remove them from the training set. This usually happens by employing some neighborhood conformity tests either on the data itself or in the latent space [38]. This type of detection, however, requires access to the training data, which is not always given (e. g., in a federated learning setting).

Other defenses try to detect poisoned models [12, 31, 38]. However, these sanitization-based defenses may be easily leveraged by an attacker who is aware of the specific defense mechanism, as they are attack-specific [27, 43].

Another perspective is to force an attacker into audible ranges, which has been successfully shown for adversarial examples [16].

7 Conclusions

While data-poisoning attacks against neural networks were already studied in the image domain, the research question of whether such attacks are also feasible in the audio domain has, so far, remained unanswered. Audio poisoning attacks have not been investigated, and recent research papers only focus on adversarial examples and their countermeasures [1].

In this paper, we tackle this problem and introduce VENOMAVE, the first data-poisoning attack against ASR systems. This is a challenging attack, given that an adversary needs to produce a *sequence* of posteriors for the target utterance, which will eventually be decoded by the ASR system into the desired sequence of words (in contrast to the image domain, where a *single* misclassification is sufficient to trigger the attack).

Using two different datasets (TIDIGITS and Speech Commands), we successfully demonstrated that such an attack is feasible in practice, and our results indicate that data-poisoning attacks are a real threat for ASR systems. We further show the stealthiness of our attack by evaluating it when the target audio waveform is played over the air in different (simulated) room settings. Similar to evasion attacks, we utilized psychoacoustic modeling to introduce less perceptible noise. Overall, with VENOMAVE, we demonstrated that data poisoning of ASR systems is a real threat and remains possible for an adversary with limited-knowledge, which makes the poisoning especially harmful for privacy-preserving training, like federated learning.

References

[1] Hadi Abdullah, Kevin Warren, Vincent Bindschaedler, Nicolas Papernot, and Patrick Traynor. SoK: The Faults in our ASRs: An Overview of Attacks against Automatic

Speech Recognition and Speaker Identification Systems. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[2] Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[3] Jont B. Allen and David A. Berkley. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 1979.

[4] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[5] Andrei Andrusenko, Aleksandr Laptev, and Ivan Medennikov. Towards a competitive end-to-end speech recognition for CHiME-6 dinner party transcription. In *Interspeech*, 2020.

[6] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International Conference on Machine Learning (ICML)*, 2018.

[7] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 2013.

[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning (ICML)*, 2012.

[9] Douglas R. Campbell, Emmanuel Vincent, and Sunit Sivasankaran. Python rir simulator, October 2021. https://github.com/sunits/rir_simulator_python, as of November 23, 2021.

[10] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *USENIX Security Symposium*, 2016.

[11] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *IEEE Security and Privacy Workshops (SPW)*, 2018.

[12] Henry Chacon, Samuel Silva, and Paul Rad. Deep learning poison data attack detection. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 971–978. IEEE, 2019.

[13] Sina Däubener, Lea Schönherr, Asja Fischer, and Dorothea Kolossa. Detecting adversarial examples for speech recognition via uncertainty quantification. In *Conference of the International Speech Communication Association (INTERSPEECH)*, 2020.

[14] Jun Du, Yan-Hui Tu, Lei Sun, Feng Ma, Hai-Kun Wang, Jia Pan, Cong Liu, Jing-Dong Chen, and Chin-Hui Lee. The ustc-iflytek system for chime-4 challenge. In *The 4th CHiME Speech Separation and Recognition Challenge*, 2016.

[15] Daniel J. Dubois, Roman Kolcun, Anna Maria Mandalari, Muhammad Talha Paracha, David Choffnes, and Hamed Haddadi. When speakers are all ears: Characterizing misactivations of iot smart speakers. In *Privacy Enhancing Technologies Symposium*, 2020.

[16] Thorsten Eisenhofer, Lea Schönherr, Joel Frank, Lars Speckemeier, Dorothea Kolossa, and Thorsten Holz. Dompteur: Taming audio adversarial examples. In *USENIX Security Symposium*, 2021.

[17] Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. In *International Conference on Learning Representations (ICLR)*, 2020.

[18] Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defensess. *Computing Research Repository (CoRR)*, abs/2012.10544, 2021.

[19] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[20] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *Computing Research Repository (CoRR)*, abs/1708.06733, 2017.

[21] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *Computing Research Repository (CoRR)*, abs/2004.00225, 2020.

[22] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[23] ISO Central Secretary. Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to 1.5 Mbits/s – Part3: Audio. Standard 11172-3, International Organization for Standardization, 1993.

[24] Tejas Jayashankar, Jonathan Le Roux, and Pierre Moulin. Detecting audio attacks on asr systems with dropout uncertainty. In *Conference of the International Speech Communication Association (INTERSPEECH)*, 2020.

[25] Naoyuki Kanda, Rintaro Ikeshita, Shota Horiguchi, Yusuke Fujita, Kenji Nagamatsu, Xiaofei Wang, Vimal Manohar, Nelson Enrique Yalta Soplin, Matthew Maciejewski, Szu-Jui Chen, et al. The hitachi/jhu chime-5 system: Advances in speech recognition for everyday home environments using multiple microphone arrays. In *The 5th CHiME Speech Separation and Recognition Challenge*, 2018.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computing Research Repository (CoRR)*, abs/1412.6980, 2014.

[27] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

[28] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *IEEE Security and Privacy Workshops (SPW)*, 2020.

[29] R. Gary Leonard and George Doddington. Tidigits ldc93s10. Linguistic Data Consortium, 1993.

[30] Juncheng B. Li, Shuhui Qu, Xinjian Li, Joseph Szurley, J. Zico Kolter, and Florian Metze. Adversarial music: Real world audio adversary against wake-word detection system. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[31] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

[32] Ivan Medennikov, Ivan Sorokin, Aleksei Romanenko, Dmitry Popov, Yuri Khokhlov, Tatiana Prisyach, Nikolay Malkovskii, Vladimir Bataev, Sergei Astapov, Maxim Korenevsky, et al. The STC system for the chime 2018 challenge. In *The 5th CHiME Speech Separation and Recognition Challenge*, 2018.

[33] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI Conference on Artificial Intelligence*, 2015.

[34] Dongyu Meng and Hao Chen. Magnet: A two-pronged defense against adversarial examples. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[35] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles A Sutton, J Doug Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter. In *Proceedings of the Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[36] H. Ney, D. Mergel, A. Noll, and A. Paeseler. A data-driven organization of the dynamic programming beam search for continuous speech recognition. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 12, pages 833–836, 1987.

[37] J. Omura. On the Viterbi decoding algorithm. *IEEE Transactions on Information Theory*, 1969.

[38] Neehar Peri, Neal Gupta, W Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P Dickerson. Deep k-nn defense against clean-label data poisoning attacks. In *European Conference on Computer Vision*, pages 55–70. Springer, 2020.

[39] Yao Qin, Nicholas Carlini, Garrison Cottrell, Ian Goodfellow, and Colin Raffel. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *International Conference on Machine Learning (ICML)*, 2019.

[40] Lea Schönherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Imperio: Robust over-the-air adversarial examples for automatic speech recognition systems. In *Annual Computer Security Applications Conference (ACSAC)*, 2020.

[41] Lea Schönherr, Maximilian Golla, Thorsten Eisenhofer, Jan Wiele, Dorothea Kolossa, and Thorsten Holz. Unacceptable, where is my privacy? exploring accidental triggers of smart speakers. *Computing Research Repository (CoRR)*, abs/2008.00508, 2020.

[42] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *Symposium on Network and Distributed System Security (NDSS)*, 2018.

[43] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning (ICML)*, 2020.

[44] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! Targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[45] Ben J Shannon and Kuldip K Paliwal. A comparative study of filter bank spacing for speech recognition. In *Microelectronic Engineering Research Conference*, 2003.

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014.

[47] Stanley Smith Stevens, John Volkmann, and Edwin B Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 1937.

[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[49] Igor Szöke, Miroslav Skácel, Ladislav Mošner, Jakub Paliesek, and Jan Černockỳ. Building and evaluation of a real room impulse response dataset. *IEEE Journal of Selected Topics in Signal Processing*, 13(4):863–876, 2019.

[50] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, 2020.

[51] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. Cocaine noodles: exploiting the gap between human and machine speech recognition. In *USENIX Security Symposium*, 2015.

[52] Lionel Sujay Vailshery. Number of digital voice assistants in use worldwide from 2019 to 2024, April 2020. https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/, as of November 23, 2021.

[53] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *Computing Research Repository (CoRR)*, abs/1804.03209, 2018.

[54] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *Proceedings of the European Conference on Artificial Intelligence*, 2012.

[55] Hiromu Yakura and Jun Sakuma. Robust audio adversarial example for a physical attack. In *International Joint Conference on Artificial Intelligence*, 2019.

[56] Wonho Yang. *Enhanced Modified Bark Spectral Distortion (EMBSD): an Objective Speech Quality Measrure Based on Audible Distortion and Cognition Model*. PhD thesis, Temple University Graduate Board, May 1999.

[57] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[58] Chen Zhu, W Ronny Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning (ICML)*, 2019.

[59] Eberhard Zwicker and Hugo Fastl. *Psychoacoustics: Facts and Models*. Springer, third edition, 2007.

A Psychoacoustic Modeling

As we discussed in the paper, recent adversarial attacks against ASR systems [39, 42] utilize psychoacoustic hearing thresholds –which describe limitations of the human auditory system– to hide modifications of the input audio signal within inaudible ranges.

By using hearing thresholds, we encourage VENOMAVE in some experiments to limit audible distortions. These thresholds define how dependencies between certain frequencies can mask, i.e., make inaudible, other parts of an audio signal. In essence, we guide VENOMAVE to hide malicious noise in these inaudible parts. At each step of the poison crafting, we scale the gradients of the poison audio signal (calculated via minimizing Equation 1) with scaling factors that limit audible distortions. Since human thresholds alone are tight, the scaling factors are allowed for differing from the thresholds by a margin of $\mathbf{\Lambda}$ (in dB). The higher $\mathbf{\Lambda}$, the more audible noise is allowed to be added by the attack.

In the following, we discuss how we compute the scaling factors. First, we compute the power spectrum of the difference $\mathbf{D}$ between the poison signal spectrum $\mathbf{\Upsilon}$ and the original signal spectrum $\mathbf{O}$ for all times t and frequencies q as the following:

$$D(t,q) = 20 \times \log_{10} \frac{|\Upsilon(t,q) - O(t,q)|}{\max_{t,q}(|O|)}, \forall t,q.$$

Then, we compute the audible difference (in dB) for all times t and frequencies q via

$$\zeta(t,q) = \mathrm{D} - \mathrm{H},$$

where $\mathbf{H}$ is the computed human hearing thresholds based on the psychoacoustic model of MPEG-1 [23]. Since the thresholds H are tight, we allow VENOMAVE to differ from the hearing thresholds by a margin of $\mathbf{\Lambda}$ (in dB). In particular, we calculate the matrix ζ^* for all times t and frequencies q as

$$\zeta^*(t,q) = \begin{cases} H(t,q) + \Lambda - D(t,q) & \text{if } H(t,q) + \Lambda \geq D(t,q) \\ 0 & \text{else} \end{cases}$$

where we clip the negative values to zero for the time-frequency bins, in which we cross the thresholds $\mathrm{H} + \Lambda$. We then normalize the matrix ζ^* to values between zero and one via

$$\hat{\zeta}(t,q) = \frac{\zeta^*(t,q) - \min_{t,q}(\zeta^*)}{\max_{t,q}(\zeta^*) - \min_{t,q}(\zeta^*)}, \forall t,q.$$

We also compute a fixed scaling factor by normalizing the hearing thresholds H to values between zero and one via

$$\hat{H}(t,q) = \frac{H(t,q) - \min_{t,q}(H)}{\max_{t,q}(H) - \min_{t,q}(H)}, \forall t,q.$$

Putting the scaling factors $\hat{\zeta}$ and $\hat{\mathrm{H}}$ together, the gradient of ∇X computed via Equation 1 will be scaled as the following

$$\nabla X_{(t,q)} := \nabla X_{(t,q)} \cdot \hat{\zeta}(t,q) \cdot \hat{H}(t,q), \forall t,q.$$

This scaling happens between the DFT and the magnitude step in the computational graph.