SAMAF: Sequence-to-sequence Autoencoder Model for Audio Fingerprinting

ABRAHAM BÁEZ-SUÁREZ, ITESM, MX and Univ. of Houston, USA
NOLAN SHAH, Univ. of Houston, USA
JUAN ARTURO NOLAZCO-FLORES, ITESM, MX
SHOU-HSUAN S. HUANG, OMPRAKASH GNAWALI, and WEIDONG SHI,
Univ. of Houston, USA

Audio fingerprinting techniques were developed to index and retrieve audio samples by comparing a content-based compact signature of the audio instead of the entire audio sample, thereby reducing memory and computational expense. Different techniques have been applied to create audio fingerprints; however, with the introduction of deep learning, new data-driven unsupervised approaches are available. This article presents Sequence-to-Sequence Autoencoder Model for Audio Fingerprinting (SAMAF), which improved hash generation through a novel loss function composed of terms: Mean Square Error, minimizing the reconstruction error; Hash Loss, minimizing the distance between similar hashes and encouraging clustering; and Bitwise Entropy Loss, minimizing the variation inside the clusters. The performance of the model was assessed with a subset of VoxCeleb1 dataset, a“speech in-the-wild” dataset. Furthermore, the model was compared against three baselines: Dejavu, a Shazam-like algorithm; Robust Audio Fingerprinting System (RAFS), a Bit Error Rate (BER) methodology robust to time-frequency distortions and coding/decoding transformations; and Panako, a constellation-based algorithm adding time-frequency distortion resilience. Extensive empirical evidence showed that our approach outperformed all the baselines in the audio identification task and other classification tasks related to the attributes of the audio signal with an economical hash size of either 128 or 256 bits for one second of audio.

CCS Concepts: • **Computing methodologies → Machine learning**; **Unsupervised learning**; *Learning paradigms*; *Dimensionality reduction and manifold learning*;

Additional Key Words and Phrases: Deep learning, sequence-to-sequence autoencoder, audio fingerprinting, audio identification

A. Baez-Suarez and N. Shah contributed equally to this work.

This work has been funded by the Department of Homeland Security (DHS) under Grant No. D15PC00185 and in part by the Mexican National Council for Science and Technology (CONACYT) scholarship 328083, and also in part by the North Atlantic Treaty Organization (NATO) Science for Peace and Security Program under Grant No. G4919. This work has also been supported by the AWS Cloud Credits for Research (https://aws.amazon.com/research-credits/) and the University of Houston I2C Lab (https://i2c.cs.uh.edu/). All statements of facts, opinions, and/or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of the sponsors.

Authors' addresses: A. Báez-Suárez, Computer Science Department, ITESM, Av. Eugenio Garza Sada 2501 Sur, Tecnológico, Monterrey, Nuevo León, 64849, MX, Univ. of Houston, Computer Science Department, 4800 Calhoun Rd, Houston, Texas; email: basuam@gmail.com; N. Shah, Univ. of Houston, Computer Science Department, 4800 Calhoun Rd, Houston, Texas; email: nshah10@uh.edu; J. A. Nolazco-Flores, Computer Science Department, ITESM, Av. Eugenio Garza Sada 2501 Sur, Tecnológico, Monterrey, Nuevo León, 64849, MX; email: jnolazco@itesm.mx; S.-H. S. Huang, O. Gnawali, and W. Shi, Univ. of Houston, 4800 Calhoun Rd, Houston, Texas; emails: {s_huang, gnawali}@cs.uh.edu, wshi3@central.uh.edu.

1551-6857/2020/05-ART43 $15.00
https://doi.org/10.1145/3380828

ACM Reference format:
Abraham Báez-Suárez, Nolan Shah, Juan Arturo Nolazco-Flores, Shou-Hsuan S. Huang, Omprakash Gnawali, and Weidong Shi. 2020. SAMAF: Sequence-to-sequence Autoencoder Model for Audio Fingerprinting. *ACM Trans. Multimedia Comput. Commun. Appl.* 16, 2, Article 43 (May 2020), 23 pages.
https://doi.org/10.1145/3380828

1 INTRODUCTION

Audio is high dimensional. Perceptually similar audio samples could have substantial variance in their data streams. As a result, the comparison of one audio sample against a corpus of other audio samples is memory and computationally expensive, and neither efficient nor effective. Tasks such as indexing & retrieval, audio identification, integrity verification, music information retrieval, scene recognition, and piracy monitoring may benefit from audio fingerprinting.

An audio fingerprint is a content-based compact signature (often a hash) that summarizes essential information about the audio sample and commonly links audio samples to their corresponding metadata (e.g., sex, age, language, artist and song name, scene, object sound, among others). Ideally, an audio fingerprint–based system should identify similar samples regardless of compression, distortion (e.g., pitching, equalization, D/A-A/D conversion, coders), or interference (e.g., background noise in the transmission channel).

Traditional cryptographic hashing algorithms such as Message Digest 5 (MD5) or Secure Hash Algorithm 3 (SHA-3) can be used to generate a compact signature of audio samples. However, these algorithms are not robust to even slight variances or degradation of audio. A single modified bit in the data stream results in an entirely different signature. Audio tasks that rely on these algorithms for audio fingerprinting would be unable to handle cases of similar audio with compression, distortion, interference, or some other form of variance in the audio. Thus, these algorithms are not considered to be robust audio fingerprinting methods.

An audio fingerprinting system can be characterized by (1) the method by which audio fingerprints are generated, (2) the search and/or matching algorithm used to categorize similar audio samples, and (3) the storage size of audio fingerprints. The literature is focused on the different techniques capable of extracting meaningful audio fingerprints and the procedure to search and match them with a collection of audio fingerprints stored in a database. The fingerprint size is chosen to be as small as possible without compromising the performance of the system.

Shazam [Wang 2003] is an example of an audio fingerprinting system. The constellation algorithm, one of the most popular methods to extract audio fingerprints, was developed for the Music Information Retrieval (MIR) task, defined in this work as the activity to extract metadata (e.g., title, genre, singer) by analyzing the audio stream. Audio fingerprints were generated by sets of hash:time offset records, whereas the searching and matching algorithm worked with time pairs distributed into bins. A matching occurred when a significant number of time pairs were located in a bin corresponding to the track ID. The fingerprint size was a 64-bit structure—32 bits for the hash and 32 bits for the time offset and track ID. The recognition rate reported on 10K tracks (8 KHz mono, 16-bit samples) considered and additive noise and an additive noise + GSM compression cases, whereas the search time on 20K tracks was found on the order of 5–500 milliseconds.

The International Federation of the Phonographic Industry (IFPI) and the Recording Industry Association of America (RIAA) recognized the importance of audio fingerprinting systems and sought to devise a variety of metrics to evaluate them [Cano et al. 2005]. These include accuracy, reliability, robustness, granularity, security, versatility, scalability, complexity, and fragility. Among these metrics, accuracy, robustness, and granularity are the most used in the literature.

Advances in Artificial Intelligence (AI) and Machine Learning (ML) have led to new approaches in the processing of high-dimensional data like audio. Deep Learning (DL) is used to create representations with different levels of abstraction often using Neural Network (NN) techniques such as Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN). Deep learning–based architectures have been recently used towards audio synthesis, multimedia processing, and cross-modality fingerprints.

Our first contribution is an unsupervised deep learning framework for generating audio fingerprints (**SAMAF**). It addresses the audio identification task through a Sequence-to-Sequence Autoencoder (SA) model composed of two linked Recurrent Neural Networks (RNN). The first RNN is the Encoder, which maps the input data into a vector representation of fixed dimensionality, and the second is the Decoder, which maps the representation back to the original input data. Our second contribution is a novel loss function that, while minimizing the SA reconstruction error, creates a hash (fingerprint) space where similar hashes are as close as possible starting with a rough composition and refining with bitwise adjustment. Our third and last contribution is to provide experimental results that show a robust audio fingerprinting system with an economical hash size (128/256 bits per second of audio) and support for a small granularity (1–6 seconds), both of which differentiate this model from other approaches.

Experimental results confirmed that our methodology is robust to transformations in speech (VoxCeleb [Nagrani et al. 2017] dataset) achieving the highest accuracy of 66.56% while the runner-up, Dejavu, [Drevo 2013], an open-source Shazam-like implementation, achieved an accuracy of 47.32%; the other two baselines, Robust Audio Fingerprinting System (RAFS) [Haitsma and Kalker 2002], a Bit Error Rate (BER) methodology robust to time-frequency distortions and coding/decoding transformations, and Panako [Six and Leman 2014], a constellation-based algorithm adding time-frequency distortion resilience, achieved accuracies of 19.82% and 23.28%, respectively.

The rest of the article is organized as follows: In Section 2, related work in audio and media fingerprinting is given. In Section 3, a brief review of deep learning is given and the SAMAF framework is broken down. In Section 4, important implementation details and experimental results are presented. Finally, in Section 5, relevant findings are summarized.

2 RELATED WORK

A survey on audio fingerprinting [Cano et al. 2005] described the general framework that nearly all techniques can be characterized by. Generally, these techniques have been motivated by a need to identify audio samples without regard for format, metadata, distortions such as noise, and without watermark embeddings. Improvements to resilience made systems such as Shazam [Wang 2003], Philips [Haitsma and Kalker 2002], and Microsoft [Burges et al. 2003] popular, especially for the music information retrieval task. In recent years, new methodologies have been developed on former approaches but introducing novel strategies such as Panako [Six and Leman 2014], which introduced robustness towards time-frequency distortions.

Haitsma and Kalker split the audio stream (mono stereo, 5 KHz) into frames of 0.37 second length and applied a Fast-Fourier Transformation (FFT). A 32-bit sub-fingerprint was extracted every 11.5625 milliseconds, applying 33 non-overlapping bands (300 Hz–2 KHz) per frame. The concatenation of 256 sub-fingerprints is called a fingerprint-block, with a total length of 2.96 seconds. The bit value (0 or 1) is determined to compute the energy relation between frames and bands. The search algorithm is a lookup table containing all the 32-bit sub-fingerprints, and the matching is calculated through Bit Error Rate (BER), finding a match if the value is below a threshold (0.35) and either returning an error or stating that the song is not in the database. The fingerprint size was found to be 2,767.5676 bits per second.

Table 1. Knowledge-based Methodologies for the Creation of Fingerprints in Different Types of Media

Author	Media	Task	Technique
Anguera et al. [2012]	Audio	Music Information Retrieval	MASK
Fan and Feng [2016]	Audio	Music Information Retrieval	Shazam-based
Henaff et al. [2011]	Audio	Music Information Retrieval	Constant-Q Transform
Haitsma and Kalker [2002]	Audio	Music Information Retrieval	Philips
Wang [2003]	Audio	Music Information Retrieval	Shazam
Sonnleitner and Widmer [2016]	Audio	Music Information Retrieval	Quad
Six and Leman [2014]	Audio	Music Information Retrieval	PANAKO
Özer et al. [2004]	Audio	Audio Identification	Periodicity & SVD based
Burges et al. [2003]	Audio	Audio Identification	Microsoft
Baluja and Covell [2007]	Audio	Audio Identification	Google
Gupta et al. [2010]	Audio	Piracy Monitoring	Philips-based
Hsieh et al. [2007]	Audio	Piracy Monitoring	Low-bit encoding & parity
Ouali et al. [2015]	Audio	Piracy Monitoring	Binary 2D Spectrogram Image
Ouali et al. [2015]	Video	Piracy Monitoring	V-Intensity & V-motion
Roopalakshmi and Reddy [2015]	Video	Piracy Monitoring	Geometric Distortions & MFCC
Park et al. [2015]	Images	Indexing and Retrieval	Neighbor Sensitive Hashing
Gaoy et al. [2014]	Images	Indexing and Retrieval	Data Sensitive Hashing
Gaoy et al. [2014]	Text	Indexing and Retrieval	Data Sensitive Hashing
Gaoy et al. [2014]	Data	Indexing and Retrieval	Data Sensitive Hashing

Burges et al. split the audio stream (mono stereo, 11.025 KHz) into frames of 372 milliseconds' length with half-overlap and applied a Modulated Complex Lapped Transform (MCLT) ending with 2,048 coefficients per frame. In the first layer, MCLT log magnitudes are projected to a 64-dimensional space with an Oriented Principal Component Analysis (OPCA) technique. Later, 32 of the resulting frames are concatenated to form another 2,048-dimensional vector, which is projected using a second layer. The final 64 real values vector is the audio fingerprint created from 6.138 seconds of audio. The search algorithm used a look-up table in the first phase. In the second, the"fingerprint to different clip" average Euclidean distance was normalized to unity enabling a single accept threshold for all fingerprints as part of the matching algorithm. The fingerprint size was found to be 667.3183 bits per second.

Six and Leman preprocessed the audio files with a sample rate of 8 KHz and 1-channel. They used TarsosDSP [Six et al. 2014] for the extraction of spectrogram features. Based on Shazam, peaks were obtained finding the local maxima and then validating that they were also the maximum within a tile with dimensions $\Delta T \times \Delta F$. They handled the time stretching through event triplets same as symbolic fingerprinting [Arzt et al. 2012] and added pitch-shift resilient property through a Constant-Q Transform [Brown and Puckette 1992]. The matching algorithm was similar to Shazam but heavily modified to allow time and frequency distortions. Their approach was tested up to 10% distortion with queries of length 20, 40, and 60 seconds. The fingerprint size was found to be 1,024 bits per second.

In the literature, fingerprints are not limited to audio, but other types of media such as video, images, text, and data. Algorithms that create media fingerprints with hand-crafted features and no meta-data are classified as knowledge-based methods, while the ones capable of creating their own features and being driven by data are classified as Machine Learning (ML) methods. Table 1 classifies the recent work of knowledge-based methodologies based on the type of media and the

Table 2. Machine Learning Methodologies, Focused on Deep Learning, for the Creation of Fingerprints in Different Types of Media

Author	Media	Task	Approach	Technique
Kereliuk et al. [2015]	Audio	Music Information Retrieval	Supervised	CNN
Li et al. [2016]	Audio	Music Information Retrieval	Unsupervised	Kohonen
Petetin et al. [2015]	Audio	Scene Recognition	Both	DBN & DNN
Amiriparian et al. [2017]	Audio	Scene Recognition	Unsupervised	SA
Gu et al. [2016]	Video	Indexing and Retrieval	Supervised	CNN & RNN
Nguyen and Do [2016]	Images	Indexing and Retrieval	Supervised	DNN
Lai et al. [2015]	Images	Indexing and Retrieval	Supervised	CNN
Hinton and Salakhutdinov [2006]	Images	Indexing and Retrieval	Unsupervised	RBM
Liong et al. [2015]	Images	Indexing and Retrieval	Both	DNN
Cao et al. [2016]	Images-Text	Indexing and Retrieval	Supervised	CNN & RNN
Salakhutdinov and Hinton [2009]	Text	Indexing and Retrieval	Unsupervised	RBM

Abbreviations: DNN - Deep Neural Network || DBN - Deep Belief Network || RBM - Restricted Boltzmann Machine || CNN - Convolutional Neural Network || RNN - Recurrent Neural Network || SA - Sequence-to-Sequence Autoencoder.

task they are solving, and Table 2 classifies the recent work of Machine Learning methodologies, with focus on Deep Learning (DL), making the distinction between supervised and unsupervised approaches. It is noted that knowledge-based methods are well developed for the creation of audio fingerprints, but other media types are mostly unexplored. In the case of DL, fingerprints for images are more studied than audio. Hence, this work builds a DL strategy for the creation of robust audio fingerprints resilient to compression, time-frequency distortions, and other transformations such as echo, low/high pass filters, or noise.

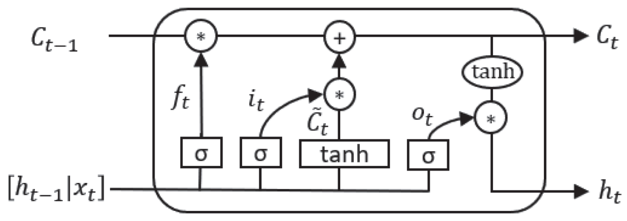
Although the methodologies found in Li et al. [2016], Petetin et al. [2015], and Amiriparian et al. [2017] are unsupervised, the first uses the unsupervised Kohonen Self-Organized Map (SOM) as a classifier, the second uses the unsupervised Deep Belief Network (DBN) as pre-training for a DNN classifier, and the third uses the unsupervised SA model to create features to be input into a multi-layer perceptron for scene classification. Unlike the proposed framework, none of these works use an unsupervised approach specifically for the generation of audio fingerprints.

3 DEEP UNSUPERVISED AUDIO FINGERPRINTING

3.1 Deep Learning Review

Deep Neural Networks (DNNs) are feedforward networks made of multiple processing layers that learn to represent data in multiple levels of abstraction. The backpropagation algorithm feeds back the network, updating its internal parameters to compute the representation in each layer from the representation in the previous layer. This allows for the discovery of intricate structures in large datasets [LeCun et al. 2015]. DNNs have improved the state-of-the-art in speech recognition [Dahl et al. 2012; Hinton et al. 2012] and object detection [Szegedy et al. 2013], among other domains. Convolutional Neural Networks (CNNs) are a particular type of feedforward network that is much easier to train and generalizes much better than networks with fully connected layers. This type of network excels at object recognition classifying 1.2M high-resolution images (ImageNet LSVRC-2010 [Deng et al. 2009]) into 1K different classes with a 37.5% error rate with the architecture AlexNet [Krizhevsky et al. 2012].

Although DNNs and CNNs have shown tremendous flexibility and power in abstraction, they have the limitation that inputs and outputs must be encoded with a fixed dimensionality. In dynamic systems, where the structure of the data changes over time, a vector of fixed dimensionality



$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}|x_t] + b_f)\\
i_t &= \sigma(W_i \cdot [h_{t-1}|x_t] + b_i)\\
o_t &= \sigma(W_o \cdot [h_{t-1}|x_t] + b_o)\\
\tilde{C}_t &= tanh(W_C \cdot [h_{t-1}|x_t] + b_C)\\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t\\
h_t &= o_t * tanh(C_t)
\end{aligned}
$$

Fig. 1. *Left:* LSTM cell with all its gates (forget f_t, input i_t, and output o_t), the sigmoid σ & hyperbolic tangent *tanh* functions and all the interactions. The inputs (previous cell C_{t-1} state, previous hidden h_{t-1} state, and input sequence x_t), the weights $W_{\{f,i,o,C\}}$, and the biases $b_{\{f,i,o,C\}}$ are responsible for the creation of the updated cell C_t and updated hidden h_t states. The rectangles indicate layer-wise operations and ellipses indicate point-wise operations. *Right:* Mathematical definition of the gates and their interactions with the inputs to create the updated cell and updated hidden states.

encodes neither the current information nor the sequential relationship within the data. This type of data is called sequential data and can be found in audio, video, and text samples. Recurrent Neural Networks (RNNs) are networks capable of processing sequential data. Their hidden neurons form a directed cycle consisting of a hidden state h_t at time step t that processes variable-length sequences $\mathbf{x} = (x_t, x_{t+1}, \ldots, x_T)$ and functions as internal memory that captures dynamic temporal information. The hidden state h_t is updated at time step t by $h_t = f(h_{t-1}, x_t)$, where f is a non-linear activation function. Even though RNNs are robust dynamic systems, training them to learn long-term dependencies is challenging due to the backpropagated gradients either growing or shrinking at each time step. Therefore, over many time steps, they typically explode or vanish [Bengio et al. 1994]. Long-Short Term Memory (LSTM) cells [Hochreiter and Schmidhuber 1997] provided a solution by incorporating memory units, which allow the network to learn when to forget, update, or output previous hidden states given new information. As a result, remembering long-term dependencies is the default behavior. Additional depth can be added to LSTM cells by stacking them on top of each other using the output of the cell in the previous layer (ℓ-1) as the input to the cell in the current layer ℓ. Figure 1 illustrates the composition of an LSTM cell that requires three inputs: previous cell C_{t-1} and previous hidden h_{t-1} states, and input sequence x_t; and returns two outputs: updated cell C_t and updated hidden h_t states; Weights $W_{\{f,i,o,C\}}$ and biases $b_{\{f,i,o,C\}}$ need to be considered. The forget f_t gate keeps/removes information from the previous cell state C_{t-1}, then the input i_t gate decides which information is updated from the cell state candidate $\tilde{C}_t$; the two previous interactions are added to form the updated cell state C_t. Finally, the output o_t gate decides which information from the updated cell state C_t will create the updated hidden state h_t. The two types of functions presented in the illustration are the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, which maps the data in the range $[0, 1]$, and the hyperbolic tangent function $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1$, which maps the data in the range $[-1, 1]$. The rectangles indicate layer-wise operations and ellipses indicate point-wise operations.

Although RNNs enable the processing of sequential data, they tend to have difficulties with length differences between input and output or cases of complicated and non-monotonic relationships. Sequence-to-Sequence (S2S) models [Cho et al. 2014; Sutskever et al. 2014] were developed to support this functionality, first applied in the machine translation task. The model consisted of two RNNs trained jointly to maximize the conditional probability of the target sequence given a source sequence. The first RNN is the Encoder, which learns how to encode a variable-length sequence into a fixed-length vector representation; and the second RNN is the Decoder, which learns how to decode a given fixed-length vector representation back into a variable-length sequence. When the

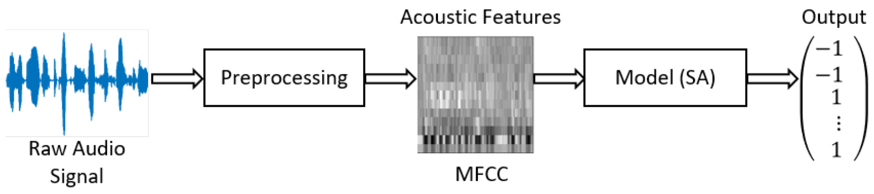


Fig. 2. The general framework starts with a raw audio signal that is normalized through a preprocessing step. The MFCC features are extracted from the normalized signal and then input to the Sequence-to-Sequence Autoencoder (SA) model, which generates the audio fingerprint (hash representation).

RNN Encoder and RNN Decoder are trained jointly to minimize the reconstruction error, the new model is called a Sequence-to-Sequence Autoencoder (SA). These types of models make minimal assumptions on the sequence structure. Therefore, they are suitable for unsupervised approaches, which remove the need for manual annotations.

3.2 SAMAF: Sequence-to-Sequence Autoencoder Model for Audio Fingerprinting

The SAMAF framework to generate audio fingerprints has similarities with classical approaches [Cano et al. 2005]. First, the raw audio signal r was preprocessed $\mathbf{r_{pp}}$ and the amplitude of the new signal $\mathbf{r_{pp}} = (r_{pp_1}, \ldots r_{pp_N}) \in I\!R^N$ was normalized following Equation (1). Then, acoustic features $\mathbf{X}$, Mel-Frequency Cepstral Coefficients (MFCCs), were extracted from the normalized signal r_{norm}. Finally, the Sequence-to-Sequence Autoencoder (SA) model processed the MFCC features and outputted the binary representation $\{1, -1\}$ of the audio fingerprint $\mathbf{H}$. An overview of our framework is found in Figure 2, while implementation technical details are located in Section 4.2.

$$r_{norm} = \frac{r_{pp}}{max(|r_{pp_1}|, \ldots, |r_{pp_N}|)} \tag{1}$$

The main component of our general framework is the SA model. In Chung et al. [2016] it is stated that the SA model proved to be useful in speech applications, because the model created fixed-dimension vector representations of variable-length audio segments. Furthermore, the unsupervised nature of this method enables the model to take the structure of the audio into account when building the representation, giving perceptually similar audio a vector representation nearby in the space. Hence, the SA model was chosen for the creation of audio fingerprints.

The model was trained with a set of $\mathbf{A}$ audio signals $\{\mathbf{X^a}\}_{a=1,\ldots,|A|}$ where $\mathbf{X^a}$ represents the ath audio signal and $\mathbf{X}$ denotes that the raw audio signal has been preprocessed, and the MFCC features have been extracted (window size of 25 ms and step of 10 ms) and grouped into T adjacent MFCC features with a hop of 25 MFCC features delivering I MFCC blocks. In the context of the RNN, T also refers to the number of steps (Section 3.1). Therefore, we have $\mathbf{X^a} = (x^a_1, x^a_2, \ldots, x^a_i, \ldots, x^a_I)$ where $x^a_i = (x^a_{i,1}, x^a_{i,2}, \ldots, x^a_{i,t}, x^a_{i,t+1}, \ldots, x^a_{i,T})$ and $x^a_{i,t} \in I\!R^M$ for all $1 \le i \le I$ and $1 \le t \le T$. The M-dimensionality is equal to the number of MFCC coefficients (13) and the sub-index t refers to the number of sequential MFCC features (100) contained in the ith MFCC block; one audio fingerprint is created per second of audio, ith MFCC block, and subsequent audio fingerprints are 25 ms or 25 MFCC features apart. A detailed representation of the SA model is illustrated in Figure 3, where the inputs to the RNN Encoder x^a_i are located on the bottom left side. The audio fingerprint size is given by the size of the LSTM cell state $S \in I\!R^B$, which was experimentally defined (either a vector of 128 or 256 real values). The previous state of the LSTM cell $S^a_{i,t-1} = (C^a_{i,t-1}, h^a_{i,t-1})$, composed of the previous cell $C^a_{i,t-1}$ and previous hidden $h^a_{i,t-1}$ states, interacts with the current inputs $x^a_{i,t}$ creating the updated state of the LSTM cell $S^a_{i,t} = (C^a_{i,t}, h^a_{i,t})$. When the last state is reached $S^a_{i,T} = (C^a_{i,T}, h^a_{i,T})$, the last hidden state $h^a_{i,T}$ contains information from all the RNN Encoder inputs x^a_i.

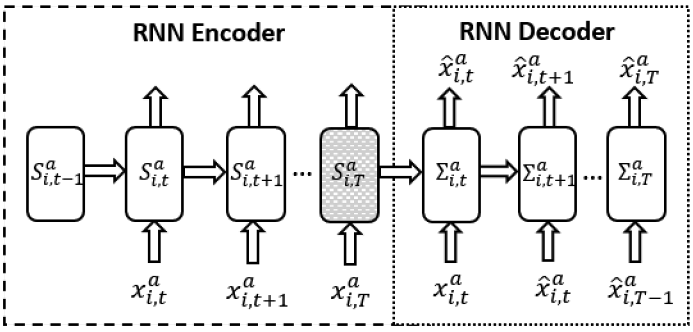


Fig. 3. The Sequence-to-Sequence Autoencoder (SA) model is composed of two RNNs: the Encoder and the Decoder. The Encoder receives the inputs x^a_i, which are combined with the LSTM cell states S^a_i. When the last state is reached $S^a_{i,T}$, it is passed to the Decoder as the initial previous state $\Sigma^a_{i,t-1}$. The output of the RNN Decoder $\hat{x}^a_i$ is a reconstructed representation of the original inputs x^a_i. The audio fingerprint (hash) is embedded into the last state $S^a_{i,T}$ of the RNN Encoder.

As a result, the last hidden state is a compact representation of the RNN Encoder inputs and is considered the feature representation for the audio fingerprint. To create the final representation of the audio fingerprint ($H^a_{i,T}$ hash), a threshold maps the last hidden state elements to its binary values $\{1, -1\}$. In Equation (2) the thresholding is formalized, stating that the last hidden state is mapped from a real space $\mathbb{R}^B$ to a Hamming space $\mathcal{H}^B$ (B-bits hash) where binary values are 1 if $h^a_{i,T}(b) \geq 0$ and -1 otherwise:

$$\begin{aligned} Threshold(h^a_{i,T}): \quad & \mathbb{R}^B \mapsto \{1,-1\}^B \\ h^a_{i,T} &= (h^a_{i,T}[1], \ldots, h^a_{i,T}[B]) \in \mathbb{R}^B \\ H^a_{i,T}(h^a_{i,T}, b) &= \begin{cases} 1, & \text{if} \quad h^a_{i,T}[b] \geq 0 \\ -1, & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

The RNN Decoder state structure $\Sigma^a_{i,t} = (\zeta^a_{i,t}, \eta^a_{i,t})$ is the same as the RNN Encoder state structure $S^a_{i,t} = (C^a_{i,t}, h^a_{i,t})$. However, the symbol variables are different to highlight the distinction in the information flow. Although Encoder and Decoder are structurally similar, their differences are that (i) the initial previous state $\Sigma^a_{i,t-1}$ of the Decoder is the last state $S^a_{i,T}$of the Encoder, (ii) the Decoder input $\hat{x}^a_{i,t}$ is the output of its previous hidden state $O(\eta^a_{i,t-1})$, commonly known as feed previous configuration, being the initial Decoder input $\hat{x}^a_{i,1}$ the same as the initial Encoder input $x^a_{i,1}$, and (iii) the learning parameters, the weights W's and biases b's, are the same for the Encoder and the Decoder.

The quality of the deep unsupervised audio fingerprint is dependent on the learned mapping to a Hamming space $f(x^a_i) : \mathbb{R}^{M \times T} \mapsto \mathcal{H}^B$. As the LSTM cell state size $S \in \mathbb{R}^B$ increases, the dimensionality of the Hamming space increases, and more information from the original audio signal can be retained by the model improving the mapping where similar audio fingerprints are nearby. As a result, the cell state size is a user-defined parameter that needs to be tuned empirically and the function to map the inputs x^a_i into the Hamming space is the loss function of the SA model.

The loss function was designed to jointly train (shared weights and biases) the RNN Encoder and RNN Decoder to generate a reconstruction sequence $\mathbf{\hat{X}^a}$ similar to the input sequence $\mathbf{X^a}$. This is done by minimizing the Mean Square Error (MSE) between the reconstruction and input; Equation (3) formalizes this idea. However, this term alone does not guarantee that similar audio fingerprints are nearby in the space. To further improve the spatial relationship between fingerprints, the Hash Loss term was added; Equation (4) formalizes it. In Equation (4), the distance matrix is calculated over the set h_T of all last hidden states given that the last hidden states $h_{i,T}$ and $h_{j,T}$ are not the same ($i \neq j$). To minimize the distance over similar last hidden states, a second set, the similarity labels set σ_T, is created, describing how strong they are related to each other. The formal definition can be found in Equation (5), where the cosine similarity metric $cos\_sim(h_{i,T}, h_{j,T}) = \frac{h_{i,T} \cdot h_{j,T}}{\|h_{i,T}\| * \|hj,T\|}$ was used to bound the range of the similarity labels between $[0, 1]$ where zero meant that the relationship did not contribute and one meant that the relationship totally contributed to the distance minimization problem. The Hash Loss was normalized by the total number of similarity labels different from zero.

$$MSE_{Loss}(\mathbf{X}) = \frac{1}{A}\sum_{a=1}^{A}\left(\mathbf{X^a} - \hat{\mathbf{X}}^\mathbf{a}\right)^2, \tag{3}$$

$$Hash_{Loss}(h_T) = \frac{\sum_{h_{i,T},h_{j,T} \in h_T, i \neq j}\left(\sigma_{ij,T}\|h_{i,T} - h_{j,T}\|^2\right)}{|\{\sigma_{ij,T} \in \sigma_T | \sigma_{ij,T} > 0\}|}, \tag{4}$$

$$\sigma_{ij,T} = \begin{cases} 1, & \text{if} \quad cos\_sim(h_{i,T}, h_{j,T}) \geq 0.55 \\ \frac{1}{0.55} * cos\_sim(h_{i,T}, h_{j,T}), & \text{if} \quad 0 \leq cos\_sim(h_{i,T}, h_{j,T}) < 0.55 \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

A Bitwise Entropy Loss was applied to fine-tune the structure of each audio fingerprint with respect to its similar audio fingerprints. Informally, $\mathbf{H_i} \subseteq \mathbb{H}$, where $\mathbb{H}$ is the set of all hashes, is defined as the set of all hashes H_j similar to H_i, including H_i. Formally, $\mathbf{H_i} = \{H_j \mid cos\_sim(H_i, H_j) \geq 0.55 \text{ where } i, j = 1, \ldots, N\}$ where N is the total number of hashes in $\mathbb{H}$. Equation (6) defines the Bitwise Entropy Loss was calculated in all the subsets $\mathbf{H_i}$ where the cardinality is greater than or equal to two. This loss term is normalized by the total number of subsets where the cardinality was greater than or equal to two. Equation (7) defines the binary entropy calculated per bit b, where each bit has two possible states $v \in \{-1, 1\}$. In information theory, the binary entropy requires the calculation of the empirical probability $P(\mathbf{H_i^{b_v}})$ when the bth bit state is v. The entropy term is normalized by the total number of elements in each subset $|\mathbf{H_i}|$ and by the number of bits B.

$$BitwiseEntropy_{Loss}(\mathbb{H}) = \frac{1}{|\{i : |\mathbf{H_i}| \geq 2\}|}\sum_{i:\mathbf{H_i} \subseteq \mathbb{H} \text{ s.t. } |\mathbf{H_i}| \geq 2} BinaryEntropy(\mathbf{H_i}), \tag{6}$$

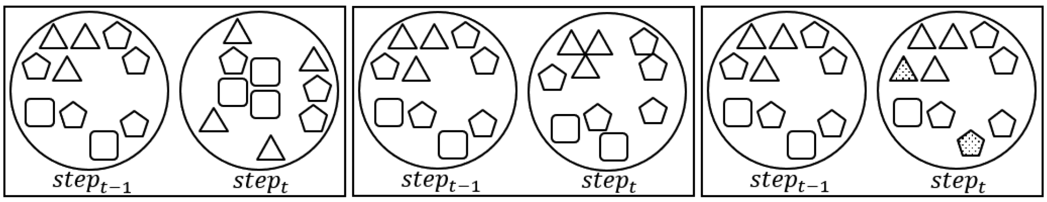
$$BinaryEntropy(\mathbf{H_i}) = \frac{1}{B} * \frac{1}{|\mathbf{H_i}|}\sum_{b=1}^{B}\sum_{v \in \{-1,1\}} -P(\mathbf{H_i^{b_v}})log_2(P(\mathbf{H_i^{b_v}})). \tag{7}$$

The final representation of the Loss function is found in Equation (8), where the three terms (i) $MSE_{Loss}(\mathbf{X})$, (ii) $Hash_{Loss}(h_t)$, and (iii) $BitwiseEntropy_{Loss}(\mathbb{H})$ are weighted by constants α, β, and γ, respectively. The weight values were set to the unity empirically:

$$Loss = \alpha * MSE_{Loss}(\mathbf{X}) + \beta * Hash_{Loss}(h_T) + \gamma * BitwiseEntropy_{Loss}(\mathbb{H}). \tag{8}$$

A visual explanation of the loss terms is presented in Figure 4 where MSE Loss minimizes the reconstruction error modifying the distribution of the audio fingerprints all around the space, the Hash Loss minimizes the distance between similar hashes bringing them together, and the Bitwise Entropy Loss minimizes the variation in the clusters of similar hashes tuning them to become more alike.

The general framework of SAMAF is presented in Algorithm 1, formalizing the steps to transform a raw audio signal r into an audio fingerprint H.



(a) MSE Loss (b) Hash Loss (c) Bitwise Entropy Loss

Fig. 4. Graphical explanation of the Loss Terms where MSE Loss minimizes the reconstruction error modifying the distribution of the audio fingerprints all around the space, the Hash Loss minimizes the distance between similar hashes bringing them together, and the Bitwise Entropy Loss minimizes the variation in the clusters of similar hashes tuning them to become more alike.

ALGORITHM 1: SAMAF - Sequence-to-Sequence Autoencoder Model for Audio Fingerprinting

```
  Data: A set R containing A raw audio signals.
  Input: Raw Audio Signal $\mathbf{R^a}$.
  Output: A Set of Audio Fingerprints $\mathbf{H^a}$ created from Raw Audio Signal $\mathbf{R^a}$.
1 repeat
2     Raw Audio Signal $\mathbf{R^a}$ is preprocessed ($\mathbf{R^a_{pp}}$)
3     Mel-Frequency Cepstral Coefficients (MFCC) $\mathbf{X^a}$ are extracted from $\mathbf{R^a_{pp}}$
4     The MFCC features are split into blocks $x^a_i$ where $i = 1, \ldots, I$.
      Each block has $T$ sequential MFCC features.
5     repeat
6         Each MFCC block $x^a_i$ is processed by the Sequence-to-Sequence Autoencoder (SA) Model
          creating an audio fingerprint $H^a_i$ per block
7     until $i$=$I$
8     A set of Hashes $\mathbf{H^a}$ are created from the MFCC features
9 until $a = A$
```

3.3 Hamming Distance Matching

Identification of a query audio sample against a collection of audio samples was carried out by comparing the number of different bits (Hamming Distance) between query hash H_q and each hash H_{c_i} in the collection. Labels were assigned to the query audio sample based on the comparison with the minimum Hamming Distance. In the case where several hashes within the collection had the same minimum Hamming Distance with the query audio sample, a voting strategy was used. Hash H_q was created following the same framework discussed in Section 3.2. As part of our research, the length of a query audio sample was varied from 1–6 seconds. Since hashes were generated on 1-second audio windows, fingerprints for samples greater than 1 second were constructed by concatenating sequential hashes. With a 0.25 second window step, a 1-second audio sample resulted in 1 hash, 2 seconds resulted in 5 concatenated sequential hashes, 3 seconds resulted in 9 concatenated sequential hashes, 4 seconds in 13 sequential hashes, 5 seconds in 17 sequential hashes, and 6 seconds in 21 sequential hashes.

4 EXPERIMENTAL RESULTS

This section covers experimental settings and results. The experiments herein show the performance of an SA model with different parameters: number of loss terms (e.g., Mean Square Error (MSE) Loss, Hash Loss, and Bitwise Entropy Loss), number of layers (either 1 or 2), and cell state size (either 128 or 256). Furthermore, to understand the robustness of each model against audio sample length (granularity), the length was varied from 1–6 seconds at a 1-second increment. Results were benchmarked against three baselines: (1) Dejavu [Drevo 2013], an open-source Shazam-like implementation, (2) RAFS [Haitsma and Kalker 2002], a Bit Error Rate (BER) methodology robust to time-frequency distortions, and (3) Panako [Six and Leman 2014], a constellation-based algorithm adding time-frequency distortion resilience. Additionally, randomly initialized SA models were evaluated to understand the intrinsic capacity of the SA model for identifying audio files and its ability to learn a mapping that boosts the identification (accuracy) performance.

4.1 VoxCeleb1 Dataset

In this work, the performance of the SA model was tested on speech in the wild with the VoxCeleb1 dataset [Nagrani et al. 2017]. Unlike most existing datasets that contain utterances under constrained conditions, the VoxCeleb dataset is a large-scale audio-visual dataset consisting of short clips of human speech extracted from interview videos uploaded to YouTube. Characteristically, the utterances contain background noise, laughter, bouts of silence, and a wide range of accents and speech. It was expected that the diversity, the unstructured nature, and the large size of the dataset should provide a challenge and reduce the potential for experimental bias contrary to more structured audio domains. The VoxCeleb1 dataset has a total of 153,516 audio files in WAV format with a 16 KHz sample rate extracted from 22,168 videos. On average, 6.93 audio files were extracted per video with a range from 1 to 239. The dataset contains speech from 1,251 different celebrities, 561 female and 690 male, spanning 36 nationalities. The total number of audio hours is 351.60 with an average segment length of 8.25 seconds, ranging from 3.26 to 144.92 seconds.

For this research, the VoxCeleb1 dataset had to be expanded adding 21 transformations[1] (TruncateSilence, Noise0.05, Echo, Reverb, HighPassFilter, LowPassFilter, Reverse, Pitch0.5, Pitch0.9, Pitch1.1, Pitch1.5, Speed0.5, Speed0.9, Speed0.95, Speed1.05, Speed1.1, Speed1.5, Tempo0.5, Tempo0.9, Tempo1.1, Tempo1.5); the transformation's parameters can be found in the Appendix. Speed, Tempo, and Pitch transformations were selected based on previous research [Sonnleitner and Widmer 2016]. Various studies have also evaluated their audio fingerprinting systems through performance against transformations [Anguera et al. 2012; Özer et al. 2004; Six and Leman 2014]. The expansion of the VoxCeleb dataset increased the amount of data significantly; hence, a subset of the originals containing audio samples between 10 and 11 seconds long was selected. The new VoxCeleb1 subset, including transformations, was composed of 141,350 audio files with 6,425 original files and 134,925 transformation files. The audio files were taken from 4,563 videos with an average of 1.41 audio samples per video and a range of 1 to 14. The dataset contains speech from 1,195 different celebrities, 537 female and 658 male, spanning 36 nationalities. The total number of audio hours is 437.73 with 18.66 hours of original audio and 419.07 hours of transformation audio. The transformation samples have an average length of 11.18 seconds.

4.2 Technical Details

As shown in Figure 2, the MFCC acoustic features were extracted from preprocessed audio signals. These features were input to the SA model where audio fingerprints were created.

Preprocessing. The audio signals were preprocessed[1] and converted to 1-channel (mono) with a 16.0 KHz sample rate. Next, the signals' amplitudes were normalized to the range $[-1, 1]$ (Equation (1)).

[1]Sound eXchange (SoX) was used to generate the transformations and to preprocess the audio signals [Bagwell 2015].

Feature Extraction. To construct the feature representation of the audio, 13 Mel-Frequency Cepstral Coefficients (MFCCs) were extracted[2] from the normalized audio using a window length of 25 ms and overlap of 10 ms. Other parameters were set to their default values.

Model. The Sequence-to-Sequence Autoencoder (SA) Model[3] is divided into an RNN Encoder and Decoder. The input vectors of the Encoder are the previous hidden and cell states with vector sizes of 128 or 256 (initialized to zero). The MFCC features have a vector size of 13. With the input vectors, the updated hidden and cell states are calculated using the weights (randomly initialized, uniformly distributed, and shared between encoder and decoder). This procedure is repeated 100 times (one second of audio is discretized to 100 steps), creating the feature representation of the audio fingerprint in the last, or T-th, step. A threshold (Equation (2)) is applied to the feature representation, which outputs the hash with its binary values $\{-1, 1\}$. To add depth to the model, the LSTM cells are stacked using the output of the LSTM in the previous layer $(\ell - 1)$ as the input to the LSTM cell in the current layer ℓ. In this research, one and two layers were used. To accelerate computation and calculate values such as the loss and gradient with statistical variation, a batch of 32 samples was used.

The Decoder's first step uses the hidden and cell states from the Encoder's last step and takes as input the same MFCC features input in the first step of the Encoder. The following steps use the Decoder's prior hidden and cell states and take as input the prior output (feed previous configuration) to compute the current step (Figure 3).

The SA model was trained with an Adagrad Optimizer using a learning rate of 0.01 without decay. The number of epochs, or the number of iterations through the entire training set, was set to 1K. However, an early stopping technique, based on training loss and accuracy curves evolution, was manually applied. The loss function (Equation (8)) was tuned with the parameters α, β, and γ equal to the unity; these values were empirically chosen. The evaluation of the model during training was based on the results of one second of audio length.

Reference Database. MySQL was used for storing the audio fingerprints. The description of the table contained the metadata (labels) of the audio file, the MFCC block ID, and the audio fingerprint (hash). This information was stored with the data types char, smallint(5) unsigned, and blob, respectively. During testing, a 176,717 entry database with a 128-bit hash size was 2.70 MB on disk and with a 256-bit hash size was 5.39 MB on disk.

Baselines. The baselines, Dejavu, RAFS, and Panako were run using open-source implementations. Dejavu, a Shazam-like algorithm, was implemented by Will Drevo,[4] and RAFS and Panako were implemented by Joren and Leman.[5]

4.3 Experiments

The experiments were developed to gain insight into the SA model audio identification performance under different parameters. The different loss function terms (MSE, Hash, and Bitwise Entropy), the cell state size $S \in \mathbb{R}^B$, the audio sample length (1–6 seconds), the number of model layers (1–2), and the model selection method during training (smallest loss vs. largest accuracy) can all have significant impact to the model's performance. Hence, the core of the experiments quantified the relationship between these variables, the structure of the hash space, and the performance on the audio identification task. Furthermore, to benchmark the SA model, the performance of these

[2]python_speech_features was used for the creation of MFCC features [Lyons 2017].
[3]TensorFlow®was used to implement the SA Model [Abadi et al. 2015].
[4]https://github.com/worldveil/dejavu.
[5]https://github.com/JorenSix/Panako.

models was compared against the performance of three baselines, Dejavu, RAFS, and Panako, using the VoxCeleb1 dataset.

The hash space was analyzed to understand the relationship between the different loss terms and the audio identification performance. Our intuition was that a structured hash space should boost the audio identification task performance. Furthermore, the analysis of the hash space gave us insight about the learning process given the unsupervised nature of the SA model.

The audio identification task was focused on recognizing variations of the original audio signal. In the case of the subset of the VoxCeleb1 dataset, the transformations are determined to be successfully recognized when its metadata (speaker, video ID, filename) is matched to its corresponding original audio signal's metadata.

The characteristics of the model such as number of layers and state size could have a significant effect on the ability of the model to construct abstract features or express complex relationships between features. Since both of these are crucial to the development of an effective state (and therefore hash) that encodes the audio signal, they may positively or negatively impact the audio identification task. To determine this impact, the experiments performed were evaluated over different numbers of layers (1–2) and state size (128 or 256).

The granularity of the system was tested from a range of 1 to 6 seconds with hashes created over a window of 1 second and a step of 0.25 second. With a granularity of 1 second, the fingerprint was constructed with one hash. With 2 seconds of audio, the fingerprint was constructed with 5 concatenated sequential hashes, 3 seconds was constructed with 9 concatenated sequential hashes, and so on. This means, for a sample audio 10.015 seconds long and a granularity of 1 second, 37 fingerprints would be created and a query against the database would be performed for each hash. For a similarly sized audio sample and granularity of 2 seconds, 33 fingerprints would be created and queried. For a 3-second audio granularity, 29 fingerprints would be created and queried. Audio identification results are based on the exhaustive creation of hashes along the entire audio signal rather than a finite number of random queries as done in other studies. With this testing methodology, results are reproducible and statistically more significant, because they represent the complete audio signal instead of random parts of it.

The model was selected based on two different strategies: smallest loss or largest accuracy. The smallest loss selection strategy selected the model with a better constructed hash space, because it minimized the reconstruction error (MSE Loss), distance between similar hashes (Hash Loss), and diversity in a cluster of similar hashes (Bitwise Entropy Loss). The largest accuracy selection strategy observed training evaluation accuracies and selected the model with the best audio identification performance. Even though the hypothesis was that a structured hash space would deliver better identification results, the unsupervised nature of the SA model meant the model did not necessarily learn features that improved performance over the audio identification task, instead favoring features to better recognize other attributes of the audio signal (e.g., speech signal of the speaker).

The samples in the dataset were split into three sets: Training (60%), Validation (20%), and Testing (20%). The subset of the VoxCeleb1 dataset (originals of length 10–11 seconds long) contains 6,425 audio files labeled as originals from 1,195 people. Twenty-one transformations were applied to the originals, resulting in 134,925 new audio files labeled as transformations. In sum, the training set contained 2,799 original files and 58,779 transformation files, and validation and test sets each contained 1,812 and 1,184 original files and 38,052 and 38,094 transformation files, respectively. In terms of hours per respective set, there were 190.74 hours of training audio including originals and transformations and 123.43 and 123.56 hours of validation and test audio, respectively, for a total of 437.73 hours. Furthermore, in terms of number of hashes, there were 2,529,518 hashes in the training set and 1,636,790 and 1,638,401 hashes in the validation and test sets, respectively, for

Table 3. Quantity of Reference Hashes Stored in the Reference Database and Quantity of Query Hashes with Respect to the Audio Length

Length	1 second	2 seconds	3 seconds	4 seconds	5 seconds	6 seconds
#References	176,717	158,265	139,813	121,361	102,909	84,457
#Queries	41,561	37,513	33,465	29,417	25,369	21,321

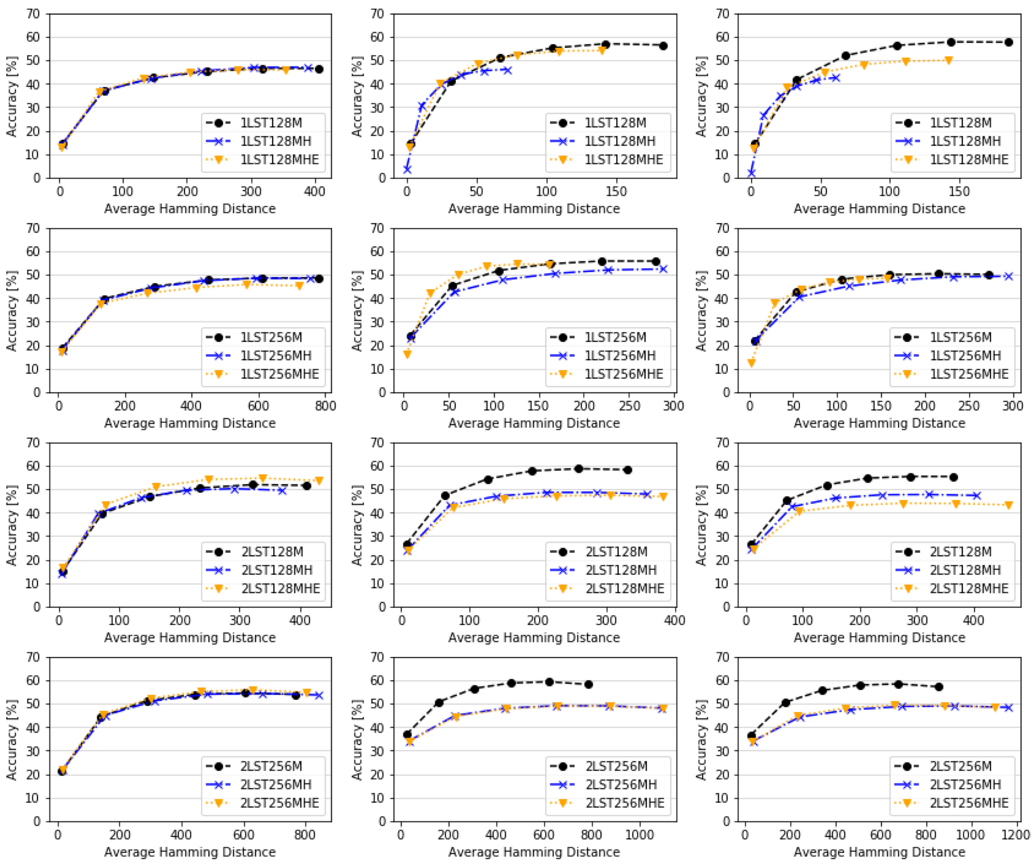
a total of 5,804,709. While all Training set hashes were used during the learning process, just 2% of the Training and Validation set hashes were used for model evaluation to speed up the learning process. The training time was about 21 days per model.

During task performance evaluations, all original hashes (training, validation, and testing sets) were stored using MySQL, forming the "Reference Database." Each query hash H_q was compared against all hashes in the Reference Database and matched with the one with the minimum Hamming Distance. Machine learning algorithms generalization performance needs to be evaluated against unseen data. Therefore, the models were evaluated with a Known set composed of hashes from the training set that were seen by the model during training and the Unknown set composed of hashes from the testing set that were never seen by the model during training. Since 36 different models were evaluated for each granularity from 1–6 seconds, the time to evaluate accuracy performance was improved by constructing the Known and Unknown sets from only 1% of training and 1% testing hashes, respectively. Table 3 displays the amount of reference and query hashes with respect to the audio length.

Three baselines were used to benchmark our methodology. **Dejavu** is a Shazam-like approach that uses the constellation algorithm. It requires 2-Channels (stereo) and a sample rate of 44.1 KHz in MP3 format. The number of hashes depends on the length of the audio signal and the number of peaks found in the frequency spectrum. **RAFS** downmixed audio files to 1-Channel with a sample rate of 16 KHz, window size of 2,048, and step size of 64. The BER was kept at 0.35 with a threshold of 2,867 bits of error. The fingerprint size was found to be 8K bits per second. **Panako**, a variation of the constellation algorithm adding time-frequency distortion resilience, preprocessed the audio files to 1-Channel with a sample rate of 16 KHz, window size of 256, and a step size of 64. The information between peaks was augmented with time stretching and pitch-shift, making it more robust to these types of transformations. The fingerprint size was found to be 32,768 bits per second. RAFS and Panako default parameters did not work for small audio excerpts (1–6 seconds). Hence, the parameters mentioned above had to be empirically determined. All the other parameters were kept at the implementation defaults. All three techniques, Dejavu, RAFS, and Panako, encoded the relationship between peaks in the hash. Our approach drastically differs in that the audio signal information was directly encoded in the hash. In addition to the three baselines, SA models with randomly initialized weights and biases were evaluated to understand the intrinsic capacity of the model and give insight into the effectiveness of the learning process (training). In our results, "Random" refers to these instances of the SA model.

4.4 Results

The outcome of the experiments describes the nature of the model providing an objective comparison of how different parameters and strategies impact performance. Furthermore, a nomenclature will be used to refer to the models; for instance, in 1LST256MHE the first characters specify the number of layers (1L/2L), the next set of characters specify the size of the hash (ST128/ST256), and last, the remaining characters specify the additive construction of the loss function where M stands for Mean Square Error Loss, H for Hash Loss, and E for Bitwise Entropy Loss.



(a) Random (b) Accuracy (c) Loss

Fig. 5. The average Hamming distance (X-axis) vs. audio identification accuracy (Y-axis) shows that the models with a structured hash space (small average Hamming distance) are not necessarily the ones with higher performance and vice versa; it depends on the parameters of the model. Random, Accuracy, and Loss indicate the method by which the models were selected. Random refers to instances of the SA model with randomly initialized weights, Accuracy refers to selection by maximal audio identification performance, and Loss refers to selection by minimal loss. [Better visualized in color].

The analysis of the hash space with respect to audio identification performance is given in Figure 5. It was expected that a structured hash space yields a lower average Hamming Distance relative to an unorganized hash space. Per our observations, the SA model was the right choice for the generation of audio fingerprints, because the Random strategy (left column) achieved good results with an unorganized hash space. Although we hypothesized that a more structured hash space would result in better performance, in reality, the audio identification performance is more correlated to the model parameters rather than the structure of the hash space. As seen on the first and second rows, the models with the smallest average Hamming Distance are not the ones with the highest performance, although the opposite can be seen on the third and final rows. Due to the unsupervised nature of the SA model, the framework parameters needed to be tuned to jointly build a structured hash space and achieve high audio identification performance. The model that accomplished both was 2LST128M-Accuracy.

Table 4. Top-five Models in the VoxCeleb1 Subset for Audio Identification and Other Classification Tasks Based on the Audio Signal Attributes

Audio Identification								**Transformations**							
Layer	**Size [bits]**	**Length [secs]**	**Focus**	**M**	**H**	**E**	**Acc [%]**	**Layer**	**Size [bits]**	**Length [secs]**	**Focus**	**M**	**H**	**E**	**Acc [%]**
2	256	5	Acc	X			59.3	2	256	6	Acc	X			66.5
2	256	4	Acc	X			58.7	2	256	5	Acc	X			66.2
2	128	5	Acc	X			58.6	2	128	6	Acc	X			66.1
2	256	5	Loss	X			58.3	2	256	6	Loss	X			65.5
2	128	6	Acc	X			58.2	2	256	5	Loss	X			65.3

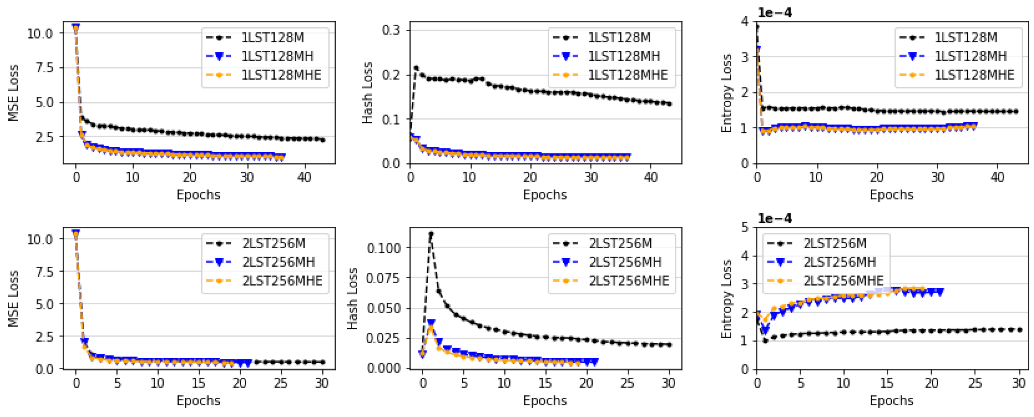
Speaker								**Video**							
Layer	**Size [bits]**	**Length [secs]**	**Focus**	**M**	**H**	**E**	**Acc [%]**	**Layer**	**Size [bits]**	**Length [secs]**	**Focus**	**M**	**H**	**E**	**Acc [%]**
2	256	5	Acc	X			59.9	2	256	5	Acc	X			59.5
2	256	4	Acc	X			59.5	2	256	4	Acc	X			59.0
2	128	5	Acc	X			59.2	2	128	5	Acc	X			58.9
2	128	6	Acc	X			58.8	2	128	6	Acc	X			58.5
2	256	5	Loss	X			58.8	2	256	5	Loss	X			58.5

Abbreviations: M - Mean Square Error Loss || H - Hash Loss || E - Bitwise Entropy Loss.

An analysis of the best SA models is given over the different attributes of the audio signal. Table 4 details the performance of the top-five models in the subset of VoxCeleb1 dataset per attribute. In general, the model 2LST256M with an audio length of 4, 5, and 6 seconds focused on highest accuracy achieved good classification performance in audio identification, transformation, speaker and video attributes. However, a model with smaller hash size, 2LST128M and focused on highest accuracy, also achieved similar performance, suggesting that this type of model can be used instead of the most accurate model if the technical implementation requires a smaller hash size. Additionally, 2-layer models with a hash size of 128/256 bits were always in the top five, suggesting that more layers enhanced the encoding of the audio signal information.

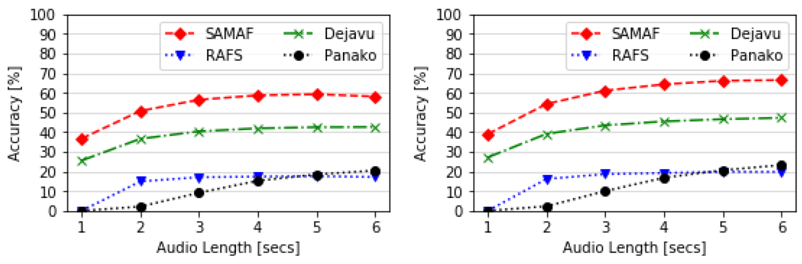
The performance of the SA models can be explained with the Loss function terms and their evolution during the training process. As shown in Figure 6, the most simple (top) and complex (bottom) models with the loss terms M/MH/MHE were able to minimize the reconstruction error (MSE Loss). As expected, the clustering term (Hash Loss) was minimized when using the loss terms MH/MHE, but M loss term followed a similar pattern only in the complex model (2LST256). This behavior implies that the Mean Square Error term performed some measure of clustering intrinsically without a term enforcing that behavior. Finally, the models that minimized variation inside the clusters (Bitwise Entropy Loss) reached, in general, high identification task accuracies.

The SA models are benchmarked against other methodologies to highlight the relative performance improvement. VoxCeleb1 subset dataset benchmark is found in Figure 7 where the best model for audio identification, 2LST256M-Acc, was used for comparison purposes. It is worth noticing that our SAMAF model outperformed all the other models no matter the audio length and the attributes of the audio signal. Its performance range was 58.19%–66.56% with a difference of at least 15% with respect to the runner-up, Dejavu. Even though Panako (2014) is a more recent model robust to time-frequency distortions, the performance was similar to RAFS (2002) in 4, 5, and 6 seconds but poor for 1, 2, and 3 seconds. In the original publications, Panako was tested with

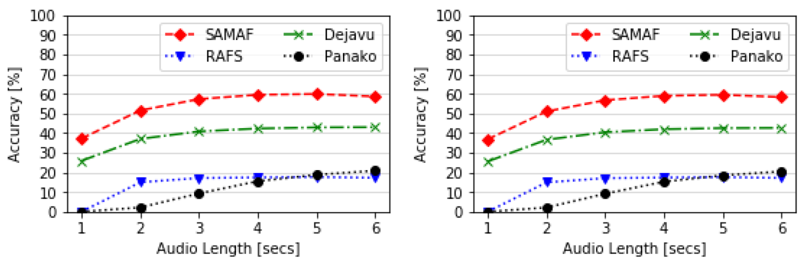


(a) Mean Square Error Loss (b) Hash Loss (c) Bitwise Entropy Loss

Fig. 6. Learning curves of the Loss function terms. The top row refers to the most simple model (1LST128) while the bottom row to the most complex model (2LST256).



(a) Audio Identification Accuracy [%] (b) Transformations Accuracy [%]



(c) Speaker Accuracy [%] (d) Video Accuracy [%]

Fig. 7. VoxCeleb1 subset dataset benchmark using the best model, 2LST256M-Acc, to report classification performance.

an audio length of 20 seconds and RAFS with an audio length of 3 seconds. This could explain experimental results.

The evaluation of an audio fingerprinting system is not fulfilled without a performance breakdown over transformations. Table 5 reports these results with the three baselines. From the data, it is clear that, on average, SAMAF outperforms the baselines. Even when comparing results for queries without any transformation (Original vs. Original), the other three baselines are unable to identify 100% of the audio samples. Extreme transformations such as Pitch0.5 and Speed0.5 cannot be handled by any method. Surprisingly, Dejavu has a higher resilience towards positive

Table 5. VoxCeleb1 Subset Benchmarked against Baselines per Transformation

Transformation	**1 second**				**2 seconds**				**3 seconds**			
	Panako	**RAFS**	**Dejavu**	**SAMAF**	**Panako**	**RAFS**	**Dejavu**	**SAMAF**	**Panako**	**RAFS**	**Dejavu**	**SAMAF**
Original	0.00%	0.00%	65.20%	**100.00%**	15.29%	23.07%	90.67%	**100.00%**	60.70%	67.02%	96.72%	**100.00%**
TruncateSilence	0.00%	0.00%	65.18%	**80.49%**	11.75%	88.51%	**90.45%**	86.38%	53.55%	89.85%	**96.63%**	89.32%
Noise0.05	0.00%	0.00%	**46.84%**	21.50%	2.09%	1.52%	**74.78%**	62.32%	10.30%	1.65%	**86.95%**	74.77%
Echo	0.00%	0.00%	**77.58%**	70.78%	1.87%	17.89%	95.65%	**97.11%**	6.32%	24.59%	98.74%	**98.86%**
Reverb	0.00%	0.00%	64.18%	**96.26%**	1.07%	57.90%	90.86%	**100.00%**	8.43%	58.08%	97.08%	**100.00%**
HighPassFilter	0.00%	0.00%	**61.58%**	24.28%	4.30%	93.99%	**89.34%**	77.58%	21.36%	94.64%	**96.50%**	87.53%
LowPassFilter	0.00%	0.00%	58.47%	**93.08%**	13.52%	73.77%	85.50%	**100.00%**	56.86%	74.68%	93.08%	**100.00%**
Reverse	0.00%	0.00%	**29.49%**	2.04%	0.00%	0.00%	**47.67%**	7.16%	0.00%	0.00%	**57.56%**	9.10%
Pitch0.5	0.00%	0.00%	**0.06%**	0.00%	0.00%	0.00%	**0.06%**	0.00%	0.00%	0.00%	0.00%	0.00%
Pitch0.9	0.00%	0.00%	0.00%	**97.22%**	0.00%	0.00%	0.00%	**100.00%**	0.07%	0.00%	0.00%	**100.00%**
Pitch1.1	0.00%	0.00%	0.06%	**99.43%**	0.00%	0.00%	0.00%	**100.00%**	0.50%	0.00%	0.00%	**100.00%**
Pitch1.5	0.00%	0.00%	0.11%	**4.03%**	0.00%	0.00%	0.19%	**34.90%**	0.00%	0.00%	0.21%	**66.52%**
Speed0.5	0.00%	0.00%	**0.03%**	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Speed0.9	0.00%	0.00%	0.00%	**14.81%**	0.00%	0.00%	0.00%	**40.19%**	0.00%	0.00%	0.00%	**49.13%**
Speed0.95	0.00%	0.00%	0.00%	**32.19%**	0.42%	0.00%	0.00%	**47.50%**	2.07%	0.00%	0.00%	**58.49%**
Speed1.05	0.00%	0.00%	0.00%	**30.01%**	0.07%	0.00%	0.00%	**49.50%**	0.08%	0.00%	0.00%	**61.99%**
Speed1.1	0.00%	0.00%	0.06%	**24.20%**	0.00%	0.00%	0.00%	**55.31%**	0.00%	0.00%	0.00%	**75.06%**
Speed1.5	0.00%	0.00%	**0.18%**	0.09%	0.00%	0.00%	**0.32%**	0.11%	0.00%	0.00%	**0.52%**	0.00%
Tempo0.5	0.00%	0.00%	**8.21%**	3.55%	0.00%	0.00%	**13.73%**	6.36%	0.00%	0.00%	**18.30%**	5.54%
Tempo0.9	0.00%	0.00%	**42.02%**	30.13%	0.11%	0.00%	**66.78%**	55.91%	0.12%	0.00%	**80.79%**	73.79%
Tempo1.1	0.00%	0.00%	**52.01%**	31.51%	0.00%	0.07%	**76.98%**	64.36%	0.00%	0.00%	**86.32%**	84.00%
Tempo1.5	0.00%	0.00%	**27.40%**	4.02%	0.00%	0.00%	**39.01%**	11.97%	0.00%	0.00%	**47.85%**	12.10%
Average	0.00%	0.00%	27.21%	**39.07%**	2.29%	16.21%	39.18%	**54.39%**	10.02%	18.66%	43.51%	**61.19%**

Transformation	**4 seconds**				**5 seconds**				**6 seconds**			
	Panako	**RAFS**	**Dejavu**	**SAMAF**	**Panako**	**RAFS**	**Dejavu**	**SAMAF**	**Panako**	**RAFS**	**Dejavu**	**SAMAF**
Original	91.22%	79.49%	98.19%	**100.00%**	97.61%	85.24%	98.84%	**100.00%**	98.03%	82.74%	98.82%	**100.00%**
TruncateSilence	82.27%	91.19%	**97.83%**	91.27%	90.20%	92.72%	**98.55%**	93.34%	93.96%	95.12%	**98.72%**	94.21%
Noise0.05	21.98%	1.57%	**92.86%**	81.26%	34.72%	1.94%	**95.55%**	84.62%	45.17%	1.77%	**96.12%**	86.83%
Echo	12.15%	24.32%	99.29%	**99.78%**	17.86%	27.18%	99.43%	**99.92%**	23.96%	28.45%	99.32%	**99.90%**
Reverb	21.58%	57.33%	98.36%	**100.00%**	36.78%	57.96%	98.94%	**100.00%**	51.87%	58.27%	99.06%	**100.00%**
HighPassFilter	43.51%	94.98%	**97.62%**	91.49%	63.21%	95.44%	**98.74%**	93.96%	74.51%	94.68%	**98.71%**	96.09%
LowPassFilter	89.92%	74.79%	96.55%	**100.00%**	97.14%	75.24%	97.68%	**100.00%**	98.03%	74.94%	98.35%	**100.00%**
Reverse	0.00%	0.00%	**63.38%**	10.65%	0.00%	0.00%	**65.38%**	11.00%	0.00%	0.00%	**70.24%**	11.86%
Pitch0.5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Pitch0.9	0.99%	0.00%	0.00%	**100.00%**	1.84%	0.00%	0.00%	**100.00%**	3.07%	0.00%	0.00%	**100.00%**
Pitch1.1	2.80%	0.00%	0.00%	**100.00%**	5.63%	0.00%	0.00%	**100.00%**	8.51%	0.00%	0.00%	**100.00%**
Pitch1.5	0.00%	0.00%	0.49%	**80.02%**	0.00%	0.00%	0.19%	**87.05%**	0.00%	0.00%	0.59%	**89.56%**
Speed0.5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Speed0.9	0.00%	0.00%	0.00%	**46.00%**	0.80%	0.00%	0.00%	**39.21%**	2.17%	0.00%	0.00%	**32.23%**
Speed0.95	3.05%	0.00%	0.00%	**70.27%**	3.72%	0.00%	0.00%	**85.46%**	5.08%	0.00%	0.00%	**92.06%**
Speed1.05	0.18%	0.00%	0.00%	**76.90%**	0.43%	0.00%	0.00%	**92.85%**	0.80%	0.00%	0.00%	**98.94%**
Speed1.1	0.00%	0.00%	0.00%	**81.55%**	0.00%	0.00%	0.00%	**81.55%**	0.00%	0.00%	0.00%	**78.86%**
Speed1.5	0.00%	0.00%	**0.51%**	0.00%	0.00%	0.00%	**0.50%**	0.26%	0.00%	0.00%	**0.47%**	0.00%
Tempo0.5	0.00%	0.00%	**22.29%**	5.74%	0.00%	0.00%	**25.60%**	5.01%	0.00%	0.00%	**28.91%**	4.73%
Tempo0.9	0.21%	0.00%	**87.32%**	80.15%	2.57%	0.00%	**91.45%**	77.13%	5.29%	0.00%	**94.47%**	73.82%
Tempo1.1	0.00%	0.10%	**92.53%**	91.40%	1.53%	0.12%	**96.40%**	94.12%	1.80%	0.00%	**97.19%**	92.65%
Tempo1.5	0.00%	0.00%	**53.86%**	9.86%	0.00%	0.00%	**58.90%**	10.94%	0.00%	0.00%	**60.00%**	12.50%
Average	16.81%	19.26%	45.50%	**64.38%**	20.64%	19.81%	46.64%	**66.20%**	23.28%	19.82%	47.32%	**66.56%**

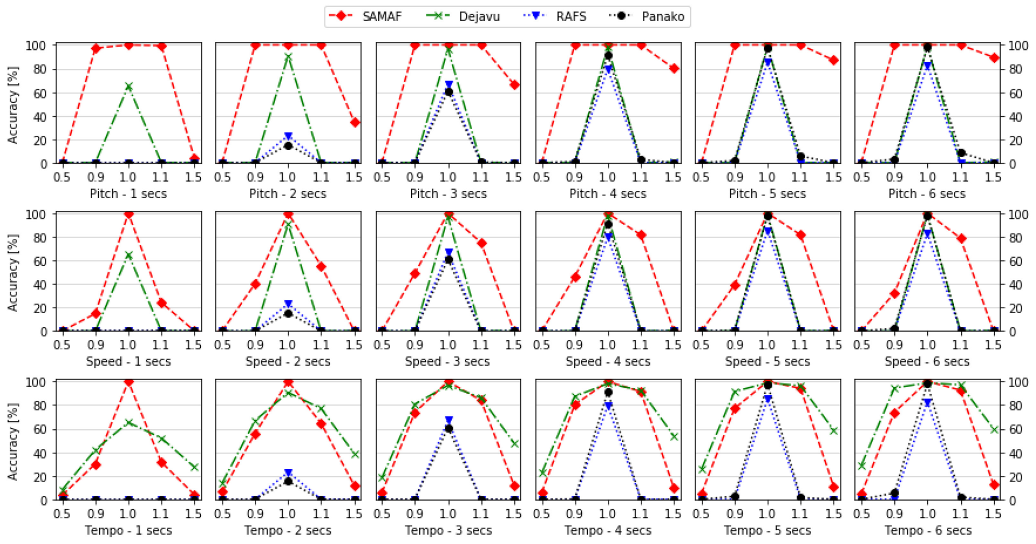


Fig. 8. Benchmark of the time-frequency transformations on the VoxCeleb1 subset dataset. Rows are Pitch, Speed, and Tempo transformations, respectively. Columns are audio sample length from 1 to 6 seconds. [Better visualized in color]. *Note:* RAFS and Panako 1-second curves are a flat line in the horizontal axis, because their performance is 0.00%.

transformations such as Speed1.5 and Tempo1.5 where the other methods failed. Dejavu also performs significantly better than all other methods in Tempo0.5, and that is the only method capable of recognizing an audio file played in reverse. Among the strong points of SAMAF, it outperformed in all Pitch and Speed distortions, excluding Speed0.5, Speed1.5, and Pitch0.5. RAFS and Panako were unable to identify queries with 1-second-long samples. However, the performance improved with larger excerpts but only for simple transformations such as Original, Truncate, Low/High Pass Filters. Neither RAFS nor Panako could handle Echo, Pitch, Speed, Tempo, and Noise transformations, although Panako showed slight improvements as audio length increased.

A visual description of the performance benchmark for time-frequency distortions is shown in Figure 8, where Pitch, Speed, and Tempo are dissected. RAFS and Panako 1-second curves are shown as a flat line over the horizontal axis because their accuracy is 0.00%. For Pitch, SAMAF outperformed the baselines and surprisingly Pitch1.5 reached almost 90% accuracy while Pitch0.5 is 0.0%. Dejavu, RAFS, and Panako only scored for the Original classification. For Speed, a similar pattern was found where SAMAF performed better than the baselines and could handle positive distortions (Speed1.1, Pitch1.5, etc.) better than negative distortions (Tempo0.5, Pitch0.9, etc.). Dejavu, RAFS, and Panako all performed for the Original with success dependent on the length of audio. However, Dejavu outperformed SAMAF on Tempo changes. Dejavu and SAMAF methods can handle small positive/negative distortions, and Dejavu is the only one able to handle with an accuracy of 60.0% an extreme transformation such as Tempo1.5. Even for the counterpart transformation, Tempo0.5, Dejavu reached about 30.0% accuracy, being the only technique resilient to variations in tempo. Although RAFS and Panako performed well only in the Original classification, Panako showed a slight performance improvement in positive and negative Tempo transformations for 5- and 6-second curves.

From a machine learning perspective, it is important to know the performance of the model for known data (training set) and unknown data (testing set) to validate the generalization capacity of the model for unseen data. Table 6 reports this metric for the most simple (top) and complex

Table 6. Most Simple (Top) and Complex (Bottom) SAMAF Models for Known vs. Unknown Comparison in the Audio Identification Task

Length	1LST128 - Acc						1LST128 - Loss					
	M		MH		MHE		M		MH		MHE	
	KWN	UNK	KWN	UNK	KWN	UNK	KWN	UNK	KWN	UNK	KWN	UNK
1 second	**14.83%**	13.36%	**3.54%**	3.38%	**13.91%**	11.71%	**15.01%**	13.23%	**2.16%**	1.95%	**13.44%**	10.64%
2 seconds	**41.76%**	40.45%	**30.98%**	30.83%	**41.30%**	38.31%	**42.52%**	40.37%	**27.67%**	25.07%	**39.66%**	36.08%
3 seconds	**51.31%**	50.78%	**39.79%**	39.54%	**49.85%**	47.00%	**52.47%**	51.20%	**35.86%**	33.02%	**46.22%**	42.96%
4 seconds	54.96%	**55.62%**	**44.30%**	43.29%	**53.25%**	50.88%	**56.55%**	55.91%	**40.01%**	37.23%	**49.37%**	46.32%
5 seconds	56.82%	**57.33%**	**46.39%**	44.22%	**54.82%**	52.42%	**57.91%**	57.69%	**42.48%**	40.01%	**51.19%**	47.07%
6 seconds	56.19%	**57.10%**	**47.00%**	44.69%	**55.18%**	52.54%	57.60%	**57.88%**	**43.71%**	40.82%	**51.39%**	47.60%
Length	2LST256 - Acc						2LST256 - Loss					
	M		MH		MHE		M		MH		MHE	
	KWN	UNK	KWN	UNK	KWN	UNK	KWN	UNK	KWN	UNK	KWN	UNK
1 second	**37.41%**	35.76%	**34.68%**	32.91%	**34.65%**	32.65%	**37.63%**	35.26%	**34.83%**	32.88%	**34.67%**	32.50%
2 seconds	**50.85%**	50.75%	**45.95%**	43.38%	**45.86%**	42.48%	**51.07%**	49.50%	**45.35%**	42.61%	**46.08%**	42.79%
3 seconds	**56.57%**	56.48%	**48.98%**	46.45%	**49.15%**	45.65%	**56.19%**	54.76%	**48.65%**	45.59%	**49.52%**	46.03%
4 seconds	58.70%	**58.90%**	**50.18%**	47.56%	**50.53%**	46.59%	**58.41%**	57.21%	**49.97%**	46.93%	**50.81%**	47.09%
5 seconds	59.05%	**59.73%**	**50.04%**	47.44%	**50.37%**	46.68%	**58.92%**	57.56%	**50.17%**	47.27%	**50.75%**	46.94%
6 seconds	57.99%	**58.49%**	**49.07%**	46.74%	**49.35%**	46.01%	**57.60%**	56.64%	**49.40%**	46.77%	**49.55%**	46.55%

(bottom) models where the difference between known and unknown data is not significant, meaning that the model is not overfitting. However, it cannot be stated that is generalizing well either. Among the different loss terms, Mean Square Error (M) was the one achieving a similar performance for known and unknown data, while MH and MHE are biased towards the known data. These results indicate that adding constraints to the hash space, in fact, harms the generalization performance and that a regularization term might alleviate this issue.

Finally, a detailed view of the performance of all the models can be found in the Appendix section, where the audio identification and transformations accuracy scores are exhaustively recorded with the full combination of the different parameters, such as number of layers, hash size, strategy, loss function terms, and audio length.

5 CONCLUSIONS

In this work, a Sequence-to-Sequence Autoencoder Model for Audio Fingerprinting (SAMAF) was developed to improve the creation of audio fingerprints through a new loss function made up of three different terms: Mean Square Error (M), Hash Loss (H), and Bitwise Entropy Loss (E). The evaluation of the model was done over a subset of the VoxCeleb1 data set. The performance of the SA model was benchmarked against three baselines: Dejavu (a Shazam-like algorithm), RAFS (a Bit Error Rate (BER) algorithm), and Panako (variation of the constellation algorithm adding time-frequency distortion resilience). Furthermore, the hash space vs. audio identification performance curves and the learning curves (training process) offered additional insight into the behavior and performance of the model.

As found in this study, a structured hash space is related to the performance of the model but not necessarily for better or worse. As a matter of fact, the selection and combination of different model parameters must be considered. Nevertheless, the model with a structured hash space delivering the best audio identification performance was composed of two LSTM layers, a 256-bit hash size, and was selected during training with maximal accuracy strategy.

The capacity of the models was tested through the audio identification task, but also the attributes an audio signal is composed of were considered to perform additional classification evaluations. It was discovered that the unsupervised nature of the SA model, the number of layers, and the hash size significantly affected the classification performance. Therefore, parameter selection could increase/decrease the classification performance per attribute, also affecting the classification scores of the audio identification task. An analysis of the learning curves from training showed the link between the progression of each loss term and audio identification performance. It was discovered that those models capable of minimizing the variation inside clusters (Bitwise Entropy Loss term) were likely to achieve the best identification task performance.

The audio identification task was performed in a subset of the VoxCeleb1 dataset. Our methodology achieved the highest score of 59.32%, while Dejavu highest score was 42.66%, RAFS 17.61%, and Panako 20.49%. The SAMAF model consistently outperformed the baselines in all the classification tasks: Audio Identification, Transformation, Speaker and Video accuracy. A detailed description of the model performance per transformation was given in Table 5 and in Figure 8 where SAMAF's best model (2LST256M-Acc), on average, performed better than the other methodologies. However, the runner-up, Dejavu, was the only method capable of either recognizing audio played in reverse or handling extreme transformations such as Speed1.5 and Tempo1.5. RAFS and Panako did not work for excerpts 1-second long or for Pitch, Speed, and Tempo transformations. Last, experimental results concluded that (1) all baselines identified audio samples under simple transformations but SAMAF did better for complex transformations; and (2) each methodology had a different tolerance to sample length over the task of audio identification.

From a Machine Learning perspective, the analysis of Known and Unknown data contributed to the proper assessment of the generalization capacity of the model. Even though the models did not overfit, they did not generalize well. As a matter of fact, adding more constraints to the creation of hashes (MH/MHE) diminished the generalization performance slightly.

ACKNOWLEDGMENTS

The authors acknowledge the theoretical contributions of Dr. Lei Xu and Dr. Lin Chen and the technical contributions of Dr. Zhimin Gao, M.Sc. Keshav Kasichainula, and Mr. Daniel Sitonic.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *Software*. Retrieved from https://www.tensorflow.org/. Version 1.13.0.

Shahin Amiriparian, Michael Freitag, Nicholas Cummins, and Björn Schuller. 2017. Sequence to sequence autoencoders for unsupervised representation learning from audio. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE'17)*.

Xavier Anguera, Antonio Garzon, and Tomasz Adamek. 2012. MASK: Robust local features for audio fingerprinting. In *Proceedings of the International Conference on Multimedia and Expo (ICME'12)*. 455–460.

Andreas Arzt, Sebastian Böck, and Gerhard Widmer. 2012. Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of the 13th International Symposium on Music Information Retrieval (ISMIR'12)*.

Chris Bagwell. 2015. SoX—Sound eXchange. *Software*. Retrieved from http://gts.sourceforge.net/ Version 14.4.2.

Shumeet Baluja and Michele Covell. 2007. Audio fingerprinting: Combining computer vision and data stream processing. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'07)*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* 5, 2 (Mar. 1994), 157–166. DOI: https://doi.org/10.1109/72.279181

Judith C. Brown and Miller S. Puckette. 1992. An efficient algorithm for the calculation of a constant Q transform. *J. Acoust. Soc. Amer.* 92, 5 (June 1992), 2698–2701.

Christopher J. C. Burges, John C. Platt, and Soumya Jana. 2003. Distortion discriminant analysis for audio fingerprinting. *IEEE Trans. Speech Aud. Proc.* 11, 3 (May 2003), 165–174.

Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. 2005. A review of audio fingerprinting. *J. VLSI Sig. Proc. Syst. Sig. Image Vid. Technol.* 41, 3 (Nov. 2005), 271–284.

Yue Cao, Mingsheng Long, Jianmin Wang, Qiang Yang, and Philip S. Yu. 2016. Deep visual-semantic hashing for cross-modal retrieval. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. 1445–1454.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1724–1734.

Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. *Audio Word2Vec: Unsupervised Learning of Audio Segment Representations using Sequence-to-sequence Autoencoder.* Research Note. College of Electrical Engineering and Computer Science, National Taiwan University, Taipei City, Taiwan.

George E. Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans. Aud. Speech Lang. Proc.* 20, 1 (Jan. 2012), 30–42.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*. 248–255. DOI: https://doi.org/10.1109/CVPR.2009.5206848

Will Drevo. 2013. Audio Fingerprinting with Python and Numpy. *Website.* Retrieved from http://willdrevo.com/fingerprinting-and-audio-recognition-with-python/.

Yong Fan and Shuang Feng. 2016. A music identification system based on audio fingerprint. In *Proceedings of the International Conference on Applied Computing and Information Technology (ACIT'16)*. 363–367.

Jinyang Gaoy, H. V. Jagadish, Wei Lu, and Beng Chin Ooi. 2014. DSH: Data sensitive hashing for high-dimensional k-NN search. In *Proceedings of the International Conference on Management of Data (SIGMOD'14)*.

Yun Gu, Chao Ma, and Jie Yang. 2016. Supervised recurrent hashing for large scale video retrieval. In *Proceedings of the ACM on Multimedia Conference (MM'16)*. 272–276.

Vishwa Gupta, Gilles Boulianne, and Patrick Cardinal. 2010. Content-based audio copy detection using nearest-neighbor mapping. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'10)*. 261–264.

Jaap Haitsma and Ton Kalker. 2002. A highly robust audio fingerprinting system. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR'02)*.

Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. 2011. Unsupervised learning of sparse features for scalable audio classification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR'11)*. 681–686.

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel Rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Sig. Proc. Mag.* 29, 6 (Nov. 2012), 82–97.

Geoffrey E. Hinton and Ruslan Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (July 28, 2006), 504–507.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780.

Che-Jen Hsieh, Jung-Shian Li, and Cheng-Fu Hung. 2007. A robust audio fingerprinting scheme for MP3 copyright. In *Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP'07)*.

Corey Kereliuk, Bob L. Sturm, and Jan Larsen. 2015. Deep learning and music adversaries. *IEEE Trans. Multimedia* 17, 11 (Nov. 2015), 2059–2071.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)*, Vol. 1. Curran Associates Inc., Lake Tahoe, NV, 1097–1105. Retrieved from http://dl.acm.org/citation.cfm?id=2999134.2999257.

Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 3270–3278.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521 (May 28, 2015), 436–444.

Hanchao Li, Xiang Fei, Kuo-Ming Chao, Ming Yang, and Chaobo He. 2016. Towards a hybrid deep-learning method for music classification and similarity measurement. In *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'16)*.

Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep hashing for compact binary codes learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 2475–2483.

James Lyons. 2017. python_speech_features. *Software.* Retrieved from https://github.com/jameslyons/python_speech_features. Version 0.6.

A. Nagrani, J. S. Chung, and A. Zisserman. 2017. VoxCeleb: A large-scale speaker identification dataset. In *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH'17)*.

Viet-Anh Nguyen and Minh N. Do. 2016. Deep learning based supervised hashing for efficient image retrieval. In *Proceedings of the International Conference on Multimedia and Expo (ICME'16)*. 1–6.

Chahid Ouali, Pierre Dumouchel, and Vishwa Gupta. 2015. Content-based multimedia copy detection. In *Proceedings of the IEEE International Symposium on Multimedia (ISM'15)*.

Hamza Özer, Bulent Sankur, and Nasir Memon. 2004. Robust audio hashing for audio identification. In *Proceedings of the European Signal Processing Conference (EUSIPCO'04)*.

Yongjoo Park, Michael Cafarella, and Barzan Mozafari. 2015. Neighbor-sensitive hashing. *J. Proc. VLDB Endow.* 9, 3 (Nov. 2015), 144–155.

Yohan Petetin, Cyrille Laroche, and Aurélien Mayoue. 2015. Deep neural networks for audio scene recognition. In *Proceedings of the European Signal Processing Conference (EUSIPCO'15)*.

R. Roopalakshmi and G. Ram Mohana Reddy. 2015. A framework for estimating geometric distortions in video copies based on visual-audio fingerprints. *J. VLSI Sig. Proc. Syst. Sig. Image Vid. Technol.* 9, 1 (Jan. 2015), 201–210.

Ruslan Salakhutdinov and Geoffrey E. Hinton. 2009. Semantic hashing. *Int. Approx. Reas.* 50, 7 (July 2009), 969–978.

Joren Six, Olmo Cornelis, and Marc Leman. 2014. TarsosDSP, a real-time audio processing framework in Java. In *Proceedings of the 53rd Audio Engineering Society Conference (AES'14)*.

Joren Six and Marc Leman. 2014. PANAKO-A scalable acoustic fingerprinting system handling time-scale and pitch modificatoin. In *Proceedings of the Conference of the International Society for Music Information Retrieval (ISMIR'14)*.

Reinhard Sonnleitner and Gerhard Widmer. 2016. Robust quad-based audio fingerprinting. *IEEE/ACM Trans. Aud. Speech Lang. Proc.* 24, 3 (Mar. 2016), 409–421.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS'14)*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3104–3112.

Christian Szegedy, Alexander Toshev, and Dumitru Erhan. 2013. Deep neural networks for object detection. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13)*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates Inc., 553–2561. Retrieved from http://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf.

Avery Li-Chun Wang. 2003. An industrial-strength audio search algorithm. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR'03)*.

Received September 2018; revised August 2019; accepted December 2019