Morphed Speech in X-Vector-Based Speaker Verification: Vulnerability and Assessment

Project-I (CS47005) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

by

Priyanshi Dixit

(20CS30069)

Under the supervision of

Prof. K. Sreenivasa Rao

Computer Science & Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2023-24

November 3, 2023

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: November 3, 2023 (Priyanshi Dixit)
Place: Kharagpur (20CS30069)

COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Morphed Speech in X-Vector-Based Speaker Verification: Vulnerability and Assessment** " submitted by **Priyanshi Dixit** (Roll No. 20CS30069) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science & Engineering is a record of bona fide work carried out by her under my supervision and guidance during Autumn Semester, 2023-24.

Date: November 3, 2023
Place: Kharagpur

Prof. K. Sreenivasa Rao
Computer Science & Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Acknowledgments

I would like to acknowledge and give special thanks to my supervisors Professor K. Sreenivasa Rao, Department of Computer Science and Engineering, IIT Kharagpur, and my mentor Mr. Aravindra Reddy. Their immense knowledge and valuable advice carried me through all the stages of my B.Tech project work, which wouldn't have been possible without their supervision. Finally, I must express my very profound gratitude to my parents and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Priyanshi Dixit

Abstract

Name of the student: **Priyanshi Dixit** Roll No: **20CS30069**

Degree for which submitted: **Bachelor of Technology**

Department: **Computer Science & Engineering**

Thesis title: **Morphed Speech in X-Vector-Based Speaker Verification: Vulnerability and Assessment**

Thesis supervisor: **Prof. K. Sreenivasa Rao**

Month and year of thesis submission: **November 3, 2023**

Speaker Verification Systems (SVS) are fundamental for ensuring secure authentication in various applications. However, their reliability is compromised by emerging threats like morphing attacks and challenges arising from gender variation. Additionally, the delicate balance of threshold settings further impacts the system's performance. Despite the existence of vulnerabilities, including those at FAR=0.1%, the intricate interplay between these factors and their collective influence on SVS accuracy remains a complex challenge.

Contents

1 Introduction

This thesis on ***Enhancing X-Vector-based Speaker Verification Systems: A Comprehensive Study on Morphing Speech, Gender Variation, Threshold Optimization, and Eventual Vulnerability Assessment*** investigates the challenges faced by Speaker Verification Systems (SVS) with a focus on morphing attacks, gender variation, and threshold optimization. Morphing attacks, which manipulate speech samples, pose a serious threat to SVS reliability by enabling unauthorized access. The research meticulously examines the impact of these attacks, revealing vulnerabilities in existing systems and exploring various levels of manipulation.

Additionally, the study delves into the complexities of gender-related challenges, analyzing how speaker gender influences SVS accuracy. Understanding these dynamics is crucial for enhancing SVS adaptability across diverse speaker profiles. Furthermore, the research emphasizes the critical role of threshold settings in balancing false acceptance and rejection rates. Through extensive experimentation, optimal configurations are identified, enhancing SVS security while ensuring user convenience.

The study recognizes SVS vulnerability at a false acceptance rate of 0.1%, highlighting the need for comprehensive solutions to fortify SVS against sophisticated attacks. By integrating insights from diverse databases, the research provides a holistic understanding of SVS vulnerabilities and proposes a robust methodology for system enhancement. This comprehensive approach ensures the resilience of SVS, making them more reliable amidst evolving cybersecurity challenges and real-world applications.

2 Existing Methodologies and Application

In the realm of speaker verification systems, X-vector-based methods have gained significant attention due to their ability to capture speaker-specific characteristics effectively. These methods rely on deep neural networks to extract discriminative speaker embeddings, which are then used for speaker verification tasks. Several existing methodologies have been explored in the context of enhancing X-vector- based speaker verification systems. These methodologies can be broadly catego- rized into the following areas:

Feature Extraction Techniques: Prior research has investigated various feature extraction techniques, such as Mel-frequency cepstral coefficients (MFCCs) and deep neural network-based embeddings. These techniques play a crucial role in extracting relevant information from speech signals, form- ing the basis for X-vector computation.

Deep Learning Architectures: Researchers have explored different deep learning architectures, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to extract speaker embed- dings. Additionally, attention mechanisms and speaker-discriminative loss functions have been incorporated to enhance the discriminative power of the extracted embed- dings.

Data Augmentation and data adaption: Data augmentation techniques, such as pitch shifting, speed perturbation, and room impulse response simulation, have been employed to augment the training data. Augmented data help in improving the robustness of X-vector models by exposing them to a diverse set of conditions. Domain adaptation methods have been explored to address the mismatch between training and testing data. Techniques like domain adversarial neural networks and adaptation using unlabeled data have been proposed to enhance the generalization ability of X-vector models.

3 Understanding X-Vector

3.1 Speaker Verification System (SVS)

A Speaker Verification System is a biometric technology designed to confirm the identity of a person based on their voice. It operates by capturing and analyzing specific vocal characteristics unique to each individual, such as pitch, tone, and speech patterns. During the enrollment process, the system records a sample of the user's voice and extracts distinctive features to create a digital voiceprint, a unique mathematical representation of their vocal traits. This voiceprint is then securely stored. When the user attempts to access a system or service, the system captures their voice again, extracts features, and compares them to the stored voiceprint. If the extracted features closely match the stored voiceprint within a certain threshold, the user's identity is verified, granting them access.

3.2 X-Vector Pipeline

The process of extracting x-vectors in speaker recognition involves several steps:

- **Features (High-Dimensional Speaker-Specific Space):** Acoustic features, like Mel-frequency cepstral coefficients (MFCCs), are extracted from the input speech signal, capturing speaker-specific characteristics in a high-dimensional space.
- **Deep Neural Network (High-Dimensional Universal Speaker Space):** Extracted features are fed into a deep neural network (DNN) which maps them into a high-dimensional universal speaker space. During training, the DNN learns to transform speaker-specific features into a more abstract, speaker-independent representation.

- **Output is X-Vector (Low-Dimensional Speaker-Specific Space):** After passing through the DNN, the high-dimensional universal speaker space representation is further transformed into a low-dimensional speaker-specific space, producing the x-vector. This x-vector is a compact, fixed-dimensional representation of the speaker's characteristics, used for speaker verification and identification.

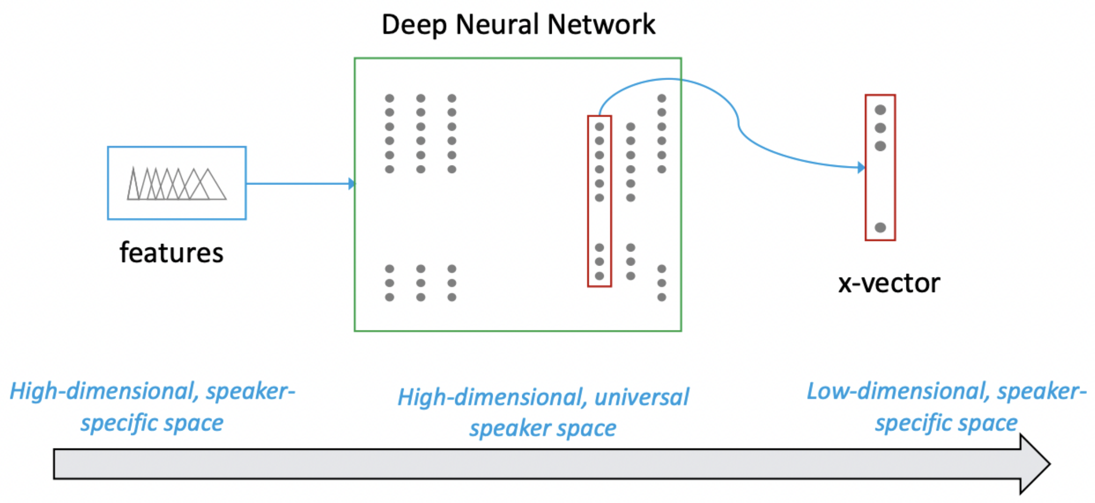

Deep Neural Network
features
x-vector
High-dimensional, speaker-specific space
High-dimensional, universal speaker space
Low-dimensional, speaker-specific space

FIGURE 1: X-vector Pipeline

3.3 X-Vector DNN

The x-vector extraction process using a Deep Neural Network (DNN) can be outlined as follows:

- **Input as Speech Feature Frames:** The raw speech signal is divided into frames, and speech feature extraction techniques, like MFCCs, are applied to each frame to obtain speech feature frames.
- **Frame-Level Layers:** Speech feature frames pass through several frame-level layers of the DNN, capturing local patterns and details from input frames.

- **Pooling Layer:** After frame-level layers, a pooling layer aggregates information from multiple frames, capturing context and temporal patterns.

- **Recording-Level Layers (1st is X-Vector):** Pooled features pass through recording-level layers. The first layer in this segment transforms the features into the x-vector, a crucial step in obtaining the final fixed-dimensional speaker representation.

- **Output Layer:** The x-vector, along with features learned from recording-level layers, is input to the output layer. This layer produces probabilities for each training speaker, indicating the likelihood that the input features belong to each speaker.

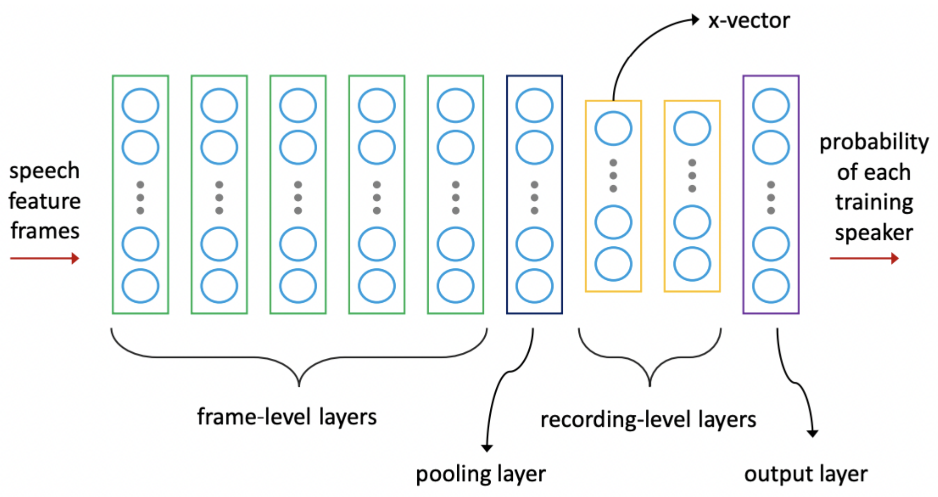


FIGURE 2: X-vector DNN

Hence, the x-vector extraction process transforms speaker-specific features into a universal speaker space using a DNN. These features are then mapped back to a low-dimensional speaker-specific space, resulting in the final x-vector representation, essential for accurate and efficient speaker recognition and verification systems.

4 Proposed method: Multilingual Speech Morphing

The proposed block diagram for Multilingual Audio Morphing (MAM) in the time domain is shown in Figure 1. The proposed morphed generation process aims to select two contributory subjects based on gender and average the signals in the time domain to generate a morphed speech signal. The proposed MAM consists of three stages 1) the selection of two contributory speakers from the database based on gender having uttered the same spoken content in two Indian languages English and Hindi 2) the average of the signals based on a percentage of spoken content of the second contributory speaker 3) finally a morphed audio is generated which is verified for both the contributory speakers.

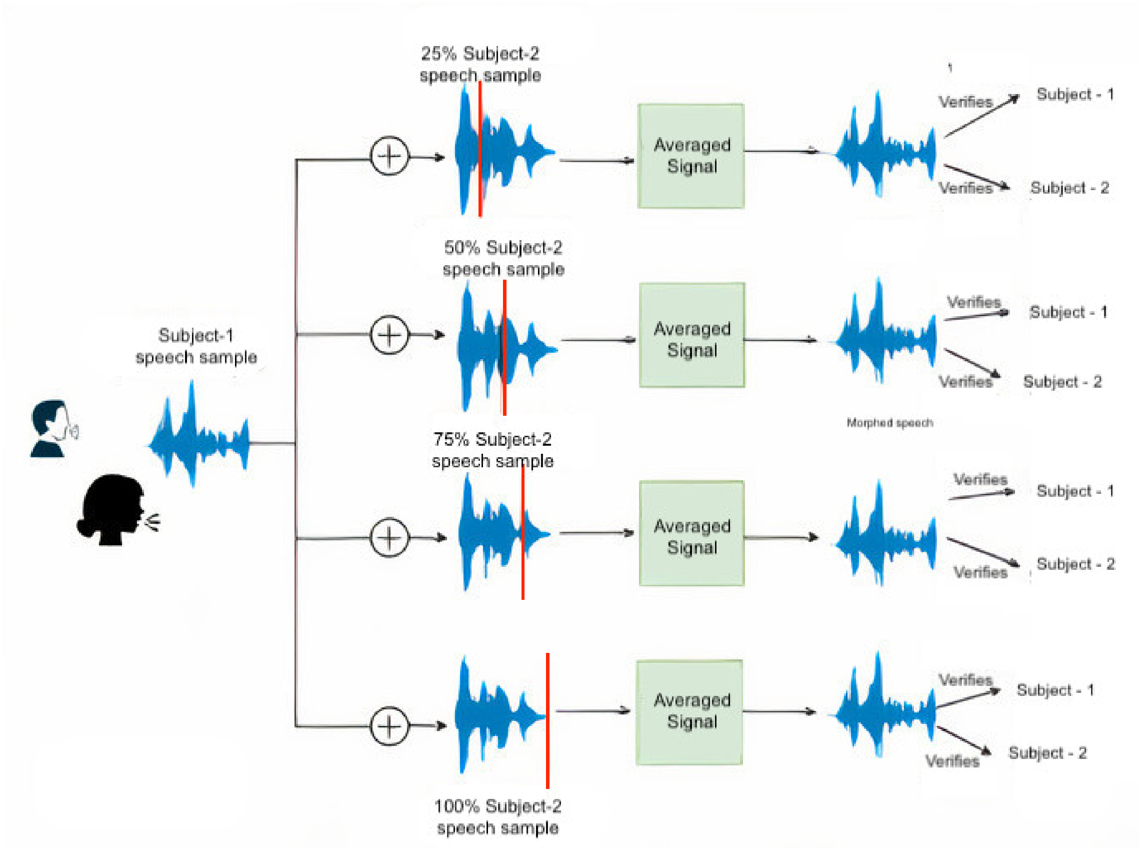


FIGURE 3: Speech Morphing by 2 speakers

4.1 *Selection of Contributory Subjects*

The proposed MAM morph generation process is based on judicious selection of two contributory speakers from the database based on gender pair i,e., male- male, male-female, female-male and female-female pairs having uttered the same spoken content. For this task, we have chosen 3 different databases namely TIMIT, SWAN multimodal database and MAVS audio-visual database. From the SMAVS database we have selected Hindi language because it contains native 50 hindi speakers.

4.2 *Generation of morphed signal*

Given two speech signals having the same spoken content from two different speakers, we name the two speakers as subject-1 and subject-2 respectively. We denote subject-1 as S_1 full spoken content and the other subject is named as S_{2i} where $i = 25\%, 50\%, 75\%, 100\%$ is the percentage of spoken content of second contributory speakers. We generate 4 different types of the morphed signal defined by:

$$S_{Morph_i} = \sum_{i \in \{25\%,50\%,75\%,100\%\}} \frac{S_1 + S_{2i}}{2} \tag{1}$$

where i is the percentage of spoken content of the second contributory subject, S_1 is the contributory speaker-1 and S_2 is second contributory speaker-2. We denote M_{25} for case-1, M_{50} for case-2, M_{75} for case-3, M_{100} for case-4 morphing for our convenience.

where the generated morphed signal is verified for both contributory subjects in all four cases.

5 Experimental Protocol

5.1 *Datasets*

We have selected 3 different publicly available databases TIMIT database [3] and SWAN multi-modal biometric dataset [4] and Multilingual Audio-visual Smartphone based dataset [5]

5.2 *TIMIT database*

The TIMIT dataset contains 10 sentences spoken by each of 630 speakers in which 438 are male and 192 are female. The TIMIT database contains total of 6300 sentences, 10 sentences are spoken by each speaker. Among 6300 sentences 2 sentences are spoken by all 630 speakers. The following two sentences are used from TIMIT database which are spoken by all speakers to perform the morphing:

- S1: She had your dark suit in greasy wash water all year.
- S2: Don't ask me to carry an oily rag like that.

Based on these sentences, we generate an exhaustive list of morphing files that resulted in a total of 443,104 files for four different types of morphing percentages as specified in the equation (1).

5.3 *Secure Access Control Wide Area Network Multi-modal Biometric Dataset (SWAN-MBD) dataset*

The SWAN-MBD is a smartphone-based audio-visual dataset that was collected in three different institutions: Norwegian University of Science and Technology (NTNU)(NO), Idiap Research Institute Switzerland, IDEMIA, and in India in 6 different sessions from 150 subjects. The SWAN-MBD consists of 5 languages such as Norwegian, Hindi, German, French and English. For our convenience, we have selected the English language from the database. All 150 speakers have spoken English as the common language. The database consists of 4 utterances, among 4 utterances 2 sentences are common. The following two sentences are used for morphing:

- S3: The limit of my card is 5000 euros.
- S4: The code is 9 8 7 6 5 4 3 2 1 0

We created morphs based on different sites i.e., databases collected at different sites (NO, FR, DE). From the utterances captured at different sites such as NO, FR, and DE we have created the following set of exhaustive morphing files as shown in Table 2

Site	No of sessions	No of speakers	No of morphed files
NTNU	6	54	$54 \times 53 \times 2 \times 4 \times 6 = 137,376$
IDIAP		60	$60 \times 59 \times 2 \times 4 \times 6 = 169,920$
UIO		52	$60 \times 59 \times 2 \times 4 \times 6 = 127,296$

Table 2: Table showing exhaustive list of number of morphed files generated at different sites for two sentences using four different types of morphing technique

Utilizing primarily English, we leveraged the diverse UIO Dataset (University of Oslo) for vital speech recognition and speaker identification research. The multilingual IDIAP Dataset (Idiap Research Institute) played a key role in language modeling and speaker verification tasks. The NTNU Dataset (Norwegian University of Science and Technology) aided regional language studies and accent recognition, while the MPH_IND Dataset specialized in Indian English and Hindi, enabling research on regional speech variations and speaker identification. Additionally, the MAVS Dataset, encompassing audio from diverse audio-visual scenes, contributed to research in sound source localization and audio-visual speech recognition, deepening our understanding of complex audio-visual interactions.

6 Proposed Methodology

6.1 *Speaker Recognition Experimentation using SpeakerRec Module*

In our methodology section, we utilized the specialized Python package, **'speechbrain/recipes/VoxCeleb/SpeakerRec'**, a component within the SpeechBrain framework. This module played a fundamental role in conducting our speaker recognition experiments, providing a solid foundation for our research endeavors. By leveraging its built-in functionalities, we streamlined our approach to various speaker recognition tasks using the VoxCeleb dataset.

The **'SpeakerRec'** module facilitated seamless data preprocessing, feature extraction, model training, and evaluation metrics implementation. Specifically, it allowed us to efficiently preprocess the VoxCeleb dataset, extracting essential features like Mel-frequency cepstral coefficients (MFCCs) vital for speaker recognition tasks.

Moreover, **‘SpeakerRec‘** offered a flexible framework, empowering us to customize different aspects of our speaker recognition system. We fine-tuned neural network architectures and experimented with diverse hyperparameters to enhance our models’ performance. This adaptability was crucial, enabling us to tailor our experiments to specific research questions and objectives.

By integrating **‘speechbrain/recipes/VoxCeleb/SpeakerRec‘** into our methodology, we ensured the robustness of our experiments while enhancing the reproducibility of our results. Its structured approach facilitated a systematic exploration of speaker recognition tasks, enhancing the reliability and credibility of our research findings.

6.2 *Configuration Setting in verification_ecapa.yaml*

In the **hparams** folder of the SpeakerRec module, the **verification_ecapa.yaml** configuration file plays a pivotal role in configuring and executing a robust Speaker Verification System utilizing the ECAPA-TDNN (ECAPA Time-Delay Neural Network) model. This section provides an in-depth exploration of the essential contents within this configuration file, detailing the parameters and settings crucial for the successful setup and operation of the Speaker Verification System.

1. Seed Initialization:
   The code initializes a random seed (1234) to ensure reproducibility of the experiments. Uses PyTorch’s `torch.manual_seed` to apply the seed.

2. Data Folders and Files:
   `data_folder_morphed:` Specifies the directory containing morphed speech data.
   `data_folder_clean:` Specifies the directory containing clean speech data.

`output_folder:` Defines the directory where the output results will be saved.

`verification_file:` Points to a text file containing information about the morphed speech data.

3. Pretrained Model Path:

   `pretrain_path:` Specifies the path to the pretrained ECAPA-TDNN model checkpoint. This model is used as a starting point for further training or evaluation.

   CSV Files:

   `train_data:` Refers to the CSV file containing training data.

   `enrol_data:` Refers to the CSV file containing enrollment data.

   `test_data:` Refers to the CSV file containing test data.

4. Data Splitting and Preprocessing:

   `split_ratio:` Defines the ratio for splitting the data into training and test sets (90% training, 10% test).

   `skip_prep:` Determines whether preprocessing steps (such as feature extraction) should be skipped (False indicates preprocessing will be performed).

5. Batch Size and Normalization:

   `batch_size:` Specifies the number of samples processed in each iteration during training and evaluation.

   `score_norm:` Sets the type of score normalization to be used (`s-norm` in this case).

   `cohort_size:` Defines the number of imposter utterances in the normalization cohort.

   `n_train_snts:` Specifies the number of training sentences used for normalization statistics.

6. Feature Parameters:

   `n_mels:` Sets the number of Mel filterbank channels used for feature extraction.

   `left_frames` and `right_frames:` Indicate the number of context frames used in feature extraction (both set to 0, possibly indicating no context frames).

   `deltas:` Specifies whether delta features are used (set to False here).

7. Dataloader Options:

   `train_dataloader_opts, enrol_dataloader_opts,` and `test_dataloader_opts` define options such as batch size for training, enrollment, and test dataloaders, respectively.

8. Feature Extraction and Normalization Modules:

   `compute_features` uses Fbank (Mel filterbank) features for audio data.

   `mean_var_norm` performs mean-variance normalization on the features.

9. Embedding Model Configuration:

   The `embedding_model` section defines the architecture and configuration of the ECAPA-TDNN neural network, which is a critical component of the speaker verification system. Here's a breakdown of its parameters:

   `!new:speechbrain.lobes.models.ECAPA_TDNN.ECAPA_TDNN:` This line instantiates a new ECAPA-TDNN model from the SpeechBrain library. ECAPA-TDNN is a type of neural network architecture specifically designed for tasks like speaker verification, where capturing temporal patterns in the input data is crucial.

   `input_size:!ref <n_mels>:`Specifies the size of the input features. In this case, it refers to the number of Mel filterbank channels (n_mels) calculated during feature extraction.

`channels: [1024, 1024, 1024, 1024, 3072]`:Describes the number of output channels for each layer in the neural network. The ECAPA-TDNN model has multiple convolutional layers. These numbers indicate how many "features" or "patterns" each layer should learn. For instance, the first layer has 1024 channels, the second layer has 1024 channels, and so on.

`kernel_sizes: [5, 3, 3, 3, 1]`:Specifies the size of convolutional kernels for each layer. Kernels are sliding windows that move across the input data to learn specific patterns. In this case, the first layer uses a kernel of size 5, the second and third layers use a kernel of size 3, the fourth layer uses a kernel of size 3, and the final layer uses a kernel of size 1.

`dilations: [1, 2, 3, 4, 1]`:Defines the dilation rate for dilated convolutions. Dilation allows the network to have a larger receptive field without increasing the number of parameters. The dilation rate determines the spacing between kernel elements. For instance, a dilation rate of 2 means there is one "hole" between each kernel element.

`attention_channels: 128:` Indicates the number of channels used in the self-attention mechanism. Self-attention allows the network to weigh the importance of different parts of the input sequence when making predictions.

`lin_neurons: 192:`Specifies the number of neurons in the linear (fully connected) layers. After the convolutional layers process the input data, the features are passed through linear layers for further processing before producing the final embeddings.

10. Pretraining Configuration:

    The `pretrainer` section is responsible for loading pretrained weights into the defined `embedding_model`. Here's what each part does:

    `!new:speechbrain.utils.parameter_transfer.Pretrainer:`
    Creates a new instance of the Pretrainer class from the SpeechBrain library. The Pretrainer class is designed to facilitate the transfer of pretrained model weights.

    `collect_in:!ref <save_folder>:`
    Specifies the directory where the code will collect necessary files, including the pretrained weights and other model-related data.

    `loadables: embedding_model:!ref <embedding_model>:`
    Indicates that the embedding_model defined earlier needs to be loaded with pretrained weights. The !ref <embedding_model>syntax refers back to the previously defined embedding_model section.

    `paths: embedding_model:!ref <pretrain_path>/embedding_model.ckpt:`
    Specifies the exact path to the pretrained weights file (embedding_model.ckpt). This file is located in the directory specified by <pretrain_path>. The !ref syntax ensures that this path is referenced correctly.

This configuration file sets up various parameters and components required for training and evaluating a Speaker Verification System. It covers data loading, preprocessing, model architecture, and training settings, providing a detailed blueprint for implementing the ECAPA-TDNN-based speaker verification system

6.3 *Speaker Verification Process Overview and Implementation of the Script*

In this subsection, we present a comprehensive overview of the `speaker_verification_cosine.py` script, which forms the core of the speaker verification process. The script adeptly manages the verification procedure by leveraging cosine similarity scores between speaker embeddings. Through a series of meticulously orchestrated steps, the script prepares essential data loaders, imports requisite libraries, and loads a pretrained model, establishing the foundation for the verification process. It then intricately computes speaker embeddings for both enrolment and test data, employing sophisticated techniques to ensure precision and reliability. Utilizing the cosine similarity metric, the script calculates verification scores for speaker pairs, a fundamental task in speaker verification systems. These scores, indicative of the similarity between speakers, are subsequently stored in an output file, encapsulating the essence of the entire verification process.

6.4 Major Libraries used:

- `os, sys:` Standard Python libraries for operating system and system-specific functionality. `torch:` PyTorch, an open-source machine learning library. `logging:` Python's built-in logging module for logging messages during program execution. `torchaudio:` PyTorch-based library for audio processing. `speechbrain:` A PyTorch-based speech processing library. `tqdm:` A library for displaying progress bars during iterations. `hyperpyyaml:` A library for reading hyperparameter files in YAML format.

6.5 Custom Functions:

a) `compute_embedding(wavs, wav_lens):`

Computes speaker embeddings from input waveforms.

`wavs:` Tensor containing speech waveforms.

`wav_lens:` Tensor containing relative lengths for each sentence.

Uses the parameters defined in `params` for feature extraction and embedding model.

Returns the computed embeddings.

b) `compute_embedding_loop(data_loader):`

Iterates through a data loader and computes embeddings for all waveforms.

Returns a dictionary where keys are segment IDs, and values are corresponding embeddings.

c) `get_verification_scores(veri_test):`

Computes positive and negative scores for speaker verification.

Utilizes cosine similarity to compute similarity scores between embeddings.

Saves the scores in a file named `morphed_score.txt`.

d) `prep_data(params):`

Prepares data loaders and defines data processing pipelines.

Loads enrolment and test data, sets up audio processing pipelines, and creates data loaders.

Returns enrolment and test data loaders.

6.6 Main Script Execution:

a) **Command Line Arguments Parsing:**

Parses command line arguments, loading hyperparameters and overrides.

b) **Data Preparation (`prep_data(params)`):**

   Loads enrolment and test data from specified CSV files.

   Defines an audio processing pipeline to load and process audio waveforms.

   Creates enrolment and test data loaders.

c) **Pretrained Model Loading:**

   Downloads and loads the pretrained embedding model using the specified pre-trainer.

   The model is loaded onto the specified device for computation.

d) **Computing Speaker Embeddings:**

   Calls `compute_embedding_loop` for enrolment and test data loaders to compute speaker embeddings.

   Embeddings are stored in dictionaries where keys are segment IDs, and values are embeddings.

e) **Speaker Verification (`get_verification_scores(veri_test)`):**

   Reads a verification file to get pairs of speaker IDs for verification.

   Computes similarity scores using cosine similarity between enrolment and test speaker embeddings.

   Writes the scores to the `morphed_score.txt` file for further analysis.

f) **Exiting the Script:**

   The script exits after computing and saving verification scores.

6.7 *Merging Session-wise Morphed Scores*

In this subsection, the Python script `merge_text.py` plays a crucial role in consolidating session-wise morphed scores obtained from different input files. The script efficiently merges the contents of four distinct session files (`session_03`, `session_04`, `session_05`, and `session_06`) containing speaker verification scores. These scores, representing the degree of similarity between speakers, are merged into a single output file named `morphed_score.txt`. This consolidation is vital for further analysis and comparison of speaker verification results across multiple sessions and experimental conditions. The script ensures seamless merging of scores while handling potential file absence scenarios with informative warnings.

6.7.1 *Segmentation of Morphed Scores Based on Speaker Categories*

In this subsection, the Python script `separate_folder.py` takes on a crucial role by categorizing the merged morphed scores into two distinct groups: 'A' and 'B'. The script reads the consolidated `morphed_score.txt` file, segregating the scores associated with the 'A' category and the 'B' category into separate lists. These categorized scores are then saved into two distinct output files: `morphed.txt` within the 'Folder_A' directory and `morphed.txt` within the 'Folder_B' directory. This segmentation process is essential for analyzing speaker verification results across specific speaker categories ('A' and 'B'), providing valuable insights into the performance variations among different speaker groups.

6.7.2 *Partitioning Morphed Scores Based on Morphing Percentage*

In this subsection, the Python script `morphed_percent.py` plays a pivotal role in refining and categorizing the merged morphed scores based on percentage values. The script reads the consolidated `morphed_score.txt` file, processes the scores, and separates them into distinct groups: '25', '50', '75', and '100'. These categorized scores are then saved into separate files: `file25.txt`, `file50.txt`, `file75.txt`, and `file100.txt` within the respective directories of both 'Folder_A' and 'Folder_B'.

This refinement and categorization process is essential for conducting in-depth analysis, allowing for detailed comparisons of speaker verification scores across various morph percentages ('25', '50', '75', and '100') within the speaker category.

6.7.3 *Gender-Based Categorization of Speaker Verification Scores*

In this subsection, the Python script `male_female.py` takes on a significant role by categorizing the refined morphed scores based on gender distinctions. The script reads the processed scores from 'Folder_A' and 'Folder_B', differentiating them into four categories: male-male (mm), male-female (mf), female-female (ff), and female-male (fm). These categorized scores are then saved into separate files: `mm.text`, `mf.text`, `ff.text`, and `fm.text` within the corresponding directories and corresponding percentage-folder under both 'Folder_A' and 'Folder_B'. This gender-based categorization process is vital for exploring the nuances in speaker verification performance concerning gender-specific variations, providing valuable insights into the effectiveness of the system across diverse gender categories.

6.7.4 *Visualization and Analysis of Speaker Verification Results*

In this subsection, the Python script `image_gen.py` plays a pivotal role in visualizing and analyzing the speaker verification results obtained from different combinations of morphing percentages and gender categories. The script processes the speaker verification scores from 'Folder_A' and 'Folder_B', categorizes them based on gender combinations (mm, mf, ff, fm), and generates comprehensive tables summarizing the verification scores. These tables provide insights into the percentage of verification scores above and below the threshold of 0.5, allowing for a detailed analysis of the system's performance across diverse morphing percentages and gender categories. Additionally, the script generates visual representations of these tables as image files, enabling clear and concise presentation of the analysis results for academic research and comparative studies.

7 Experimental Methodology

The experimental methodology undertaken in this research involved a systematic approach to speaker verification across various datasets, focusing on both text-dependent and text-independent scenarios. The experimentation was conducted across five distinct datasets: IDIAP, NTNU, MAVS, UIO, and MPH_IND. To ensure comprehensive coverage and a thorough analysis, the experiments were carried out in multiple phases, encompassing different languages and morphing percentages.

7.1 *Text-Dependent Datasets Analysis*

Initially, the research focused on text-dependent datasets, specifically IDIAP, NTNU, MAVS, UIO, and MPH_IND. The entire speaker verification process, including data preparation, feature extraction, model training, and verification score analysis, was meticulously executed for each dataset. This initial phase provided a foundational understanding of the system's performance in controlled, text-dependent scenarios.

7.2 *Text-Independent Datasets Analysis*

Following the analysis of text-dependent datasets, the experimentation extended to text-independent datasets, including IDIAP, MPH_IND, NTNU, and UIO. This phase involved adapting the verification system to handle text-independent scenarios, where the system's robustness and adaptability were rigorously evaluated. Similar to the text-dependent phase, the experiments were conducted across different languages, enhancing the research's scope and applicability.

7.3 *Language-Specific Analysis: English and Hindi*

To delve deeper into the nuances of speaker verification, the experimentation was further stratified based on languages. Specifically, the analysis was conducted separately for English and Hindi languages across the datasets. This language-specific approach allowed for a focused investigation into language-related challenges and differences in speaker verification systems' performance.

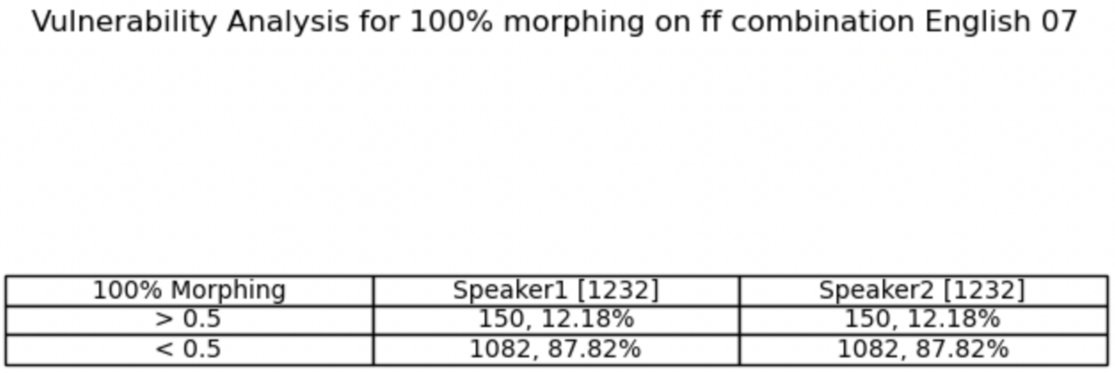
7.4 *Threshold Variation: 0.3 and 0.8*

In addition to language-specific analysis, the experiments were repeated with variations in the threshold value used for verification. Two distinct threshold values, 0.3 and 0.8, were employed to evaluate the system's sensitivity to different verification score thresholds. These threshold variations aimed to explore the system's behavior under different decision boundaries, providing insights into its performance at varying levels of stringency.

The iterative and systematic approach followed in this research, encompassing different datasets, languages, and threshold values, facilitated a comprehensive evaluation of the speaker verification system. This multi-faceted analysis not only enhanced the research's depth but also provided a holistic understanding of the system's capabilities and limitations across diverse scenarios.

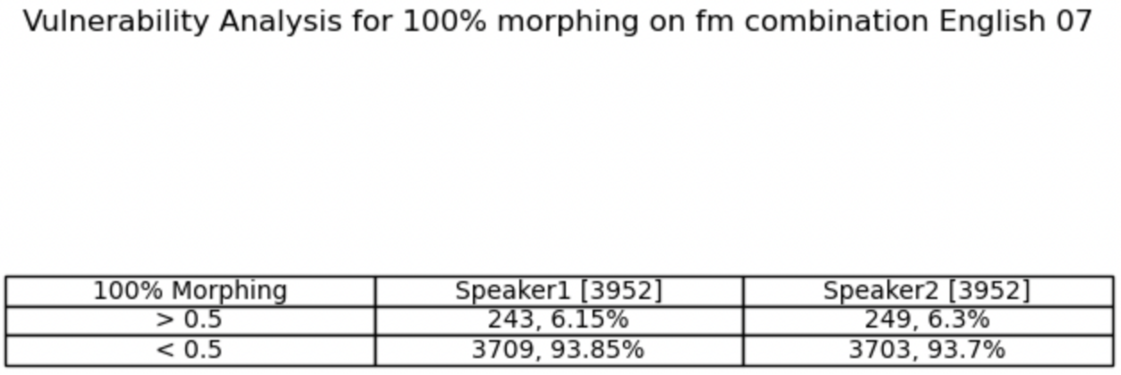
8 Experimental Results: Output Scores

In the interest of clarity and brevity, this report presents a targeted analysis of speaker verification results. We have meticulously chosen representative instances for both text-dependent (Folder_07) and text-independent (Folder_02) scenarios within the NTNU dataset. The evaluations were conducted in English, employing a morphed percentage of 100 and a threshold of 0.5.



Vulnerability Analysis for 100% morphing on ff combination English 07

100% Morphing	Speaker1 [1232]	Speaker2 [1232]
> 0.5	150, 12.18%	150, 12.18%
< 0.5	1082, 87.82%	1082, 87.82%



Vulnerability Analysis for 100% morphing on fm combination English 07

100% Morphing	Speaker1 [3952]	Speaker2 [3952]
> 0.5	243, 6.15%	249, 6.3%
< 0.5	3709, 93.85%	3703, 93.7%

Vulnerability Analysis for 100% morphing on mf combination English 07

100% Morphing	Speaker1 [3952]	Speaker2 [3952]
> 0.5	250, 6.33%	243, 6.15%
< 0.5	3702, 93.67%	3709, 93.85%

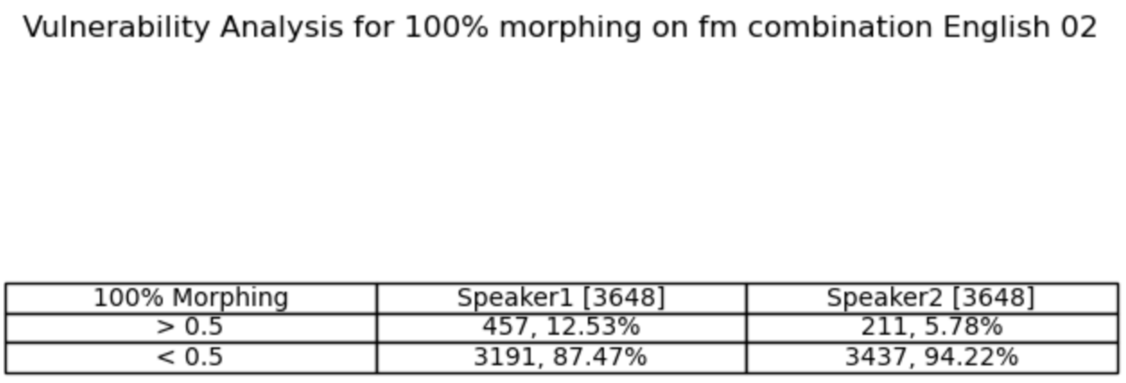
Vulnerability Analysis for 100% morphing on mm combination English 07

100% Morphing	Speaker1 [11248]	Speaker2 [11248]
> 0.5	975, 8.67%	976, 8.68%
< 0.5	10273, 91.33%	10272, 91.32%

Now for Text-Independent Case

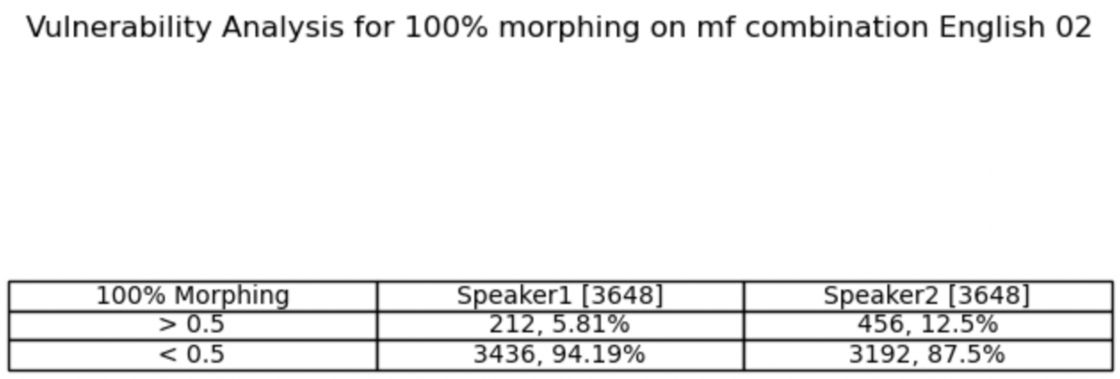
Vulnerability Analysis for 100% morphing on ff combination English 02

100% Morphing	Speaker1 [1056]	Speaker2 [1056]
> 0.5	164, 15.53%	164, 15.53%
< 0.5	892, 84.47%	892, 84.47%



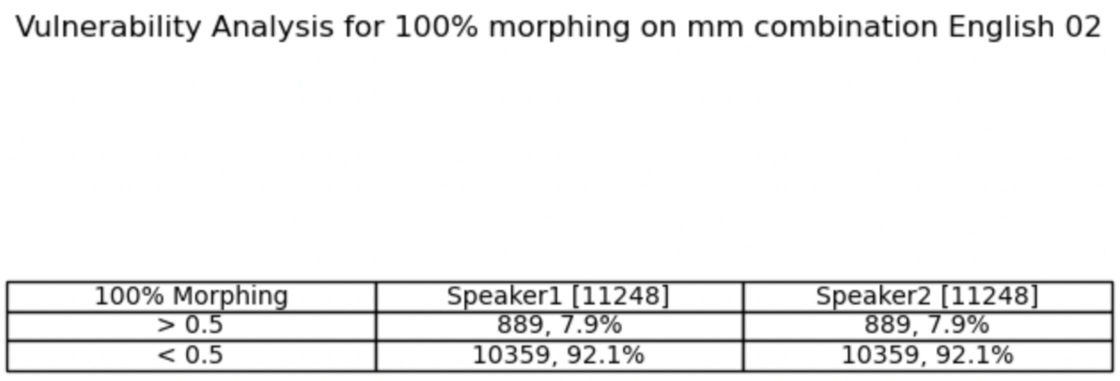
Vulnerability Analysis for 100% morphing on fm combination English 02

100% Morphing	Speaker1 [3648]	Speaker2 [3648]
> 0.5	457, 12.53%	211, 5.78%
< 0.5	3191, 87.47%	3437, 94.22%



Vulnerability Analysis for 100% morphing on mf combination English 02

100% Morphing	Speaker1 [3648]	Speaker2 [3648]
> 0.5	212, 5.81%	456, 12.5%
< 0.5	3436, 94.19%	3192, 87.5%



Vulnerability Analysis for 100% morphing on mm combination English 02

100% Morphing	Speaker1 [11248]	Speaker2 [11248]
> 0.5	889, 7.9%	889, 7.9%
< 0.5	10359, 92.1%	10359, 92.1%

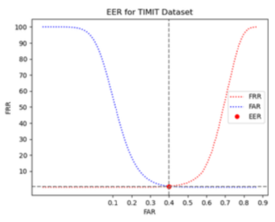
9 Vulnerability analysis

We have benchmarked a SVS system in time domain which are based on deep learning.The deep learning based SVS used in this work is the x-vector.The x-vector based ASV uses Time Delayed Neural Networks (TDNN) to excerpt 512-dimensional deep embeddings by statistically poling the varying length sentences. The x-vector based

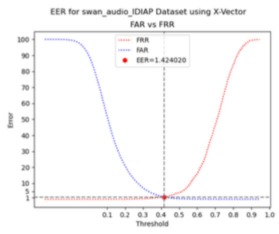
ASV systems shows lower EER for all the databases in comparison to RawNet3. The reason why we computed the EER is to show that the chosen systems are robust for speaker verification and from the table and from the plots it is evident that the chosen systems shows lesser EER and justifies our selection. The proposed morph samples are generated based on the gender pairs i,e., male-male, female-female and combined pairs. Therefore we benchmark the attack potential of MAM by comparing the verification scores computed from both the contributory subjects from the above said pairs by setting False Alarm Rate (FAR)=0.1%. The comparison scores from the Speaker Verification Systems (SVS) are computed by enrolling the morphed speech samples as attack samples and then probing the contributory pairs for both text-dependent and text-independent speech utterances. The vulnerability of the SVS can be calculated using three different types of metrices namely: Mated Morphed Presentation Match Rate (MMPMR) [6], Fully Mated Morphed Presentation Match Rate (FMMPMR) [7], Morphing At- tack Potential (MAP) [8]. The MMPMR metric is based on distinctive trials, while FMMPMR is based on discriminant (pair-wise) probe attempts of the contributory pairs. The MAP revamps the existing metrices by indicating the vulnerability as a matrix by using multiple SVS with discriminant (pair-wise) probe attempts. But still MAP does not quantify vulnerability as a single num- ber. Moreover, while calculating vulnerability, the existing metrices like MAP does not consider Failure-to-Acquire Rate (FTAR) and multiple morph gener- ation algorithms. FTAR is defined as proportion of the recognition attempts where biometric system fails to detect, identify,or acquire biometric signal of adequate quality due to failures related to user presentation or quality control. Though the enrolled speech sample (attack/morphed/bona fide) is captured in constrained conditions the probe speech sample need not be essentially captured in constrained conditions. So [9] came up with Generalised Morphing Attack 155 Potential (G-MAP) which 1) quantifies the vulnerability as a single number 2) takes into account of multiple morph generation types 3) consideration of FTAR.

SVS	Database				
	TIMIT	IDIAP	NTNU	UIO	MAVS
	EER (%)				
x-vector	0.5	1.42	1.41	2.70	7.308
RawNet3	0.5	1.38	1.43	1.45	2.43

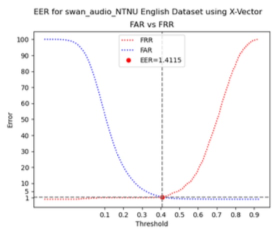
Table 3: Performance of the speaker verification systems in terms of EER (%) for different databases



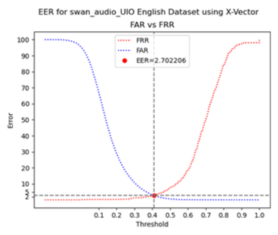
(a)



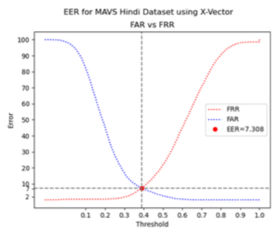
(b)



(c)



(d)



(e)

Figure 2: **(a)** EER for TIMIT database using x-vector **(b)** EER for IDIAP database usinf x-vector **(c)**EER for NTNU database using x-vector **(d)** EER for UIO database using x-vector **(e)** EER for MAVS Hindi database using x-vector

9.1 *Mathematical Expression for G-MAP*

Let $\mathbb{T}$ denotes the set of paored speech samples (in our case gender pair and also denoted as number of probe attempts), let $\mathbb{S}$ denote the set of SVS , let $\mathbb{G}$ denote the morph attach generation algorithm. Let $\mathbb{M}_a$ denote the morph speech set corresponding to $\mathbb{D}$, let τ_l similarity score from SVS (1). then G-MAP is defines as follows:

$$\begin{aligned}\text{G - MAP} = {} & \frac{1}{|\mathbb{G}|}\sum_{d}^{|\mathbb{G}|}\left(\frac{1}{|T|}\frac{1}{|M_{\mathrm{a}}|}\min_{l}\right) \\ & \times \sum_{i,j}^{|\mathbb{T}|,|\mathbb{M}_a|}\left\{\left[\left(S_{1_i}^{j} > \tau_l\right) \wedge \ldots \wedge \left(S_{k_i}^{j} > \tau_l\right)\right]\right. \\ & \left. \times \left(1 - FTAR(i,l)\right)\right\} \qquad (2)\end{aligned}$$

where *FTAR* (I,l) is the failure to acquire the probe speech sample in attempt i using SVS(l).

Since we have defined G-MAP in equation 2, which includes multiple probe attempts, multiple SVS, and morph attack generation type. G-MAP with multiple probe attempts is calculated by setting $\mathbb{G} = 1$ and $\mathbb{S} = 1$ in the equation 2. Now with FTAR=0, and similarity scores $S1_i^j$ greater than threshold τ_l, G-MAP with multiple probe attempts is equal to FMMPMR and G-MAP with multiple probe attempts and multiple SVS with $\mathbb{G} = 1$ is computed by taking minimum of vulnerability obtained from multiple SVS. Now G-MAP would quantify vulnerability as single number whereas the only GAP represents vulnerability as matrix of SVS.

9.2 *Quantitative evaluation of Vulnerability using text-dependent morphing*

In this section we evaluate the vulnerability of the two Speaker Verification Sysem x-vector for 4 different types of morphing both qualitatively and quantitatively. Since G-MAP is based on number of slots, multiple SVS, and morphing types (in this cast its 4) this will permit one to analyse the results based on a) probe attempts independent to SVS and attack speech generation category b) Multiple SVS with multiple attempts c) Finally G-MAP with a function of number of slots, multiple SVS, and corresponding different category of speech attack generation type along with FTAR.

9.3 *Vulnerability of x-vectors for SWAN-MBD database*

For SWAN-MBD database captured at different sites such IDIAP,NTNU and UIO, we have selected S3 and S4 sentences using this sentences we have created 4 types of morphing. These morphed samples are passes onto the x-vectors to get 192-dimensional embeddings from x-vector, we compute the cosine similarity and report the vulnerability for x-vector in terms of G-MAP with multiple probe attempts at FAR=0.1% for NTNU,IDIAP and UIO.G-MAP with multiple probe attempts (probe attempts=3) aims to calculate the vulnerability based on number of individual attempts i,e., each individual attempt should be a success and both the contributory speakers should be verified. Table 6 shows the quantitative analysis (G-map with multiple probe attempts) of two SVS for IDIAP database. The following points are observed from the obtained results:

- For x-vector based SVS in M_{25} only a portion of speech sample is morphed i,e., the initial frames only contains the second contributory speaker characteristics, the remaining frames does not contain the second contributory speaker characteristics, finally the frames are statistically pooled to get a single vector and latter aggregated into segment level features. So for this type of morphing, a lower vulnerability is observed.
- or M_{50} half of the morphed speech sample contains the second contributory speaker, little bit of higher vulnerability is observed than M_{25}.

- For M_{75} 2/3 of the morphed signal contains the second contributory speaker characteristics a higher vulnerability is observed than the M_{25} and M_{50}.
- For M_{100} type of morphing, the morphed signal is the average of both contributory speakers and morphed signal contains the both speaker characteristics making the x-vector based SVS making it most vulnerable to the proposed method.

Table 7 indicates G-MAP (multiple SVS and multiple probe attempts) for the four different types of morphing. Given a morph sample, it is said to be vulnerable if the multiple probe attempts must successfully circumvent the multiple SVS. Thus vulnerability is quantified as a single number by taking minimum of the SVS while taking care of FTAR. Based on the obtained results –

- The morph sample generated from M_{100} shows higher vulnerability when compared to other methods.

Table 8 shows the quantitative analysis (G-MAP with multiple probe attempts) of two SVS for NTNU database. The following points are observed from the results:

- Like in IDIAP database, M_{25}, a lower vulnerability is observed when compared to other 3 methods because of the influence of the portion of the second contributory speaker.
- highest vulnerability is observed for M_{100} when compared to other three methods.

9.4 *Quantitative evaluation of Vulnerability using text-independent morphing*

In text-independent morphing, we compare the cosine similarity between morphed samples and sentences used other than S1 and S2 from the database. Here we select two sentences randomly from the database and pass these utterances through x-vectors and to get the embeddings and report the vulnerability of SVS at FAR=0.1

SVS	**G-MAP (%) Multiple Probe Attempts**			
	Morphing Type	**Gender Pair**		
		FF (2448)	MM (7812)	Combined (10260)
x-vector (FAR=0.1%)	25%	23.76	24.74	48.54
	50%	25.79	26.78	52.57
	75%	27.17	28.62	55.79
	100%	29.12	29.34	58.46

Table 6: Vulnerability analysis for IDIAP database using G-MAP with multiple probe attempts

G-MAP (%) Multiple SVS with multiple probe attempts				
Method	**Morphing**	**Gender pair**		
	type	FF	MM	Combined
Proposed	M_{25}	23.76	24.78	48.54
	M_{50}	25.79	26.78	52.57
	M_{75}	27.17	28.62	55.79
	M_{100}	29.12	29.34	58.46

Table 7: Vulnerability analysis using G-MAP for IDIAP (Multiple probe attempts and multiple SVS) for the proposed method and VIM

SVS	**G-MAP (%) Multiple probe attempts**			
	Morphing	**Gender pair**		
	type	**FF (1672)**	**MM (16872)**	**Combined (18544)**
x-vector (FAR=0.1%)	M_{25}	23.78	14.80	38.58
	M_{50}	24.67	15.69	40.36
	M_{75}	26.79	16.58	43.37
	M_{100}	27.79	18.67	46.34

Table 8: Vulnerability ananlysis for NTNU database using G-MAP with multiple probe attempts

G-MAP (%) Multiple SVS with multiple probe attempts				
Method	**Morphing**	**Gender pair**		
	type	FF	MM	Combined
Proposed	M_{25}	22.78	16.65	39.43
	M_{50}	20.57	14.68	35.25
	M_{75}	21.77	13.83	35.60
	M_{100}	22.48	14.80	37.28
VIM				

Table 9: Vulnerability analysis using G-MAP for NTNU database (Multiple probe attempts and multiple SVS) for the proposed method and VIM

SVS	**G-MAP (%) Multiple probe attempts**			
	Morphing	**Gender pair**		
	type	**FF (2736)**	**MM (6496)**	**Combined (9292)**
x-vector (FAR=0.1%)	25%	27.2	18.93	46.13
	50%	22.59	15.45	38.04
	75%	18.56	15.28	33.84
	100%	14.18	13.34	27.52

Table 10: Vulnerability ananlysis for UIO database using G-MAP with multiple probe attempts

G-MAP (%) Multiple SVS with multiple probe attempts				
Method	**Morphing**	**Gender pair**		
	type	FF	MM	Combined
Proposed	25%	27.2	18.93	46.13
	50%	22.59	15.45	38.04
	75%	18.56	15.28	33.84
	100%	14.18	13.34	27.52
VIM				

Table 11: Vulnerability analysis using G-MAP for UIO database (Multiple probe attempts and multiple SVS) for the proposed method and VIM

10 Bibliography

[1] R. Feynman, F. Vernon Jr., The theory of a general quantum system interacting with a linear dissipative system, Annals of Physics 24 (1963) 118–173. doi:10.1016/0003-4916(63)90068-X.

G-MAP (%) Multiple SVS with multiple probe attempts				
Method	**Morphing**	**Gender pair**		
	type	FF	MM	Combined
Proposed	25%	20.78	14.75	35.53
	50%	18.57	12.68	31.25
	75%	18.77	12.83	31.6
	100%	18.48	12.80	31.28
VIM				

Table 18: Vulnerability analysis using G-MAP for NTNU database text-independent case (Multiple probe attempts and multiple SVS) for the proposed method and VIM

G-MAP (%) Multiple SVS with multiple probe attempts				
Method	**Morphing**	**Gender pair**		
	type	FF	MM	Combined
Proposed	25%	12.75	16.53	29.28
	50%	11.25	14.45	24.62
	75%	10.74	13.88	24.62
	100%	10.3	12.34	22.64
VIM				

Table 20: Vulnerability analysis using G-MAP for UIO text-independent database (Multiple probe attempts and multiple SVS) for the proposed method and VIM

SVS	**G-MAP (%) Multiple SVS with multiple probe attempts**						
	Morphing	**iPhone11**			**Samsung S8**		
	type	**FF (3168)**	**MM (14490)**	**Combined (17658)**	**FF (3168)**	**MM (14490)**	**Combined (17658)**
x-vector (FAR=0.1%)	25%	20.12	24.32	44.44	56.37	56.78	56.57
	50%	20.47	23.52	43.99	58.45	57.12	57.78
	75%	21.30	24.21	45.51	56.21	55.45	55.83
	100%	22.14	24.98	47.12	45.20	44.07	44.63

Table 22: Vulnerability analysis of MAVS database for Hindi language text-independent (Multiple probe attempts and multiple SVS) for the proposed method and VIM

SVS	**G-MAP (%) Multiple probe attempts**						
	Morphing	**iPhone11**			**Samsung S8**		
	type	**FF (3168)**	**MM (14490)**	**Combined (17658)**	**FF (3168)**	**MM (14490)**	**Combined (17658)**
x-vector (FAR=0.1%)	25%	20.12	24.32	44.44	56.37	56.78	56.57
	50%	20.47	23.52	43.99	58.45	57.12	57.78
	75%	21.30	24.21	45.51	56.21	55.45	55.83
	100%	22.14	24.98	47.12	45.20	44.07	44.63
RawNet3 (FAR=0.1%)	25%	95.21	85.12	90.16	94.23	93.02	93.62
	50%	94.18	82.93	88.55	93.21	91.20	92.20
	75%	95.40	85.21	90.30	94.56	93.09	93.82
	100%	93.20	88.58	90.89	93.05	94.87	93.96
VIM							

Table 21: Vulnerability analysis of MAVS database for Hindi language text-independent using G-MAP with multiple probe attempts

[2] P. Dirac, The lorentz transformation and absolute time, Physica 19 (1-–12) (1953) 888–896. doi:10.1016/S0031-8914(53)80099-6.

[3] J. S. Garofolo, Timit acoustic phonetic continuous speech corpus, Linguistic Data Consortium, 1993.

[4] R. Ramachandra, M. Stokkenes, A. Mohammadi, S. Venkatesh, K. Raja, P. Wasnik, E. Poiret, S. Marcel, C. Busch, Smartphone multi-modal biometric authentication: Database and evaluation, arXiv preprint arXiv:1912.02487.

[5] H. Mandalapu, P. A. Reddy, R. Ramachandra, K. S. Rao, P. Mitra, S. M Prasanna, C. Busch, Multilingual audio-visual smartphone dataset and evaluation, IEEE Access 9 (2021) 153240–153257

[6] U. Scherhag, A. Nautsch, C. Rathgeb, M. Gomez-Barrero, R. N. Veldhuis, L. Spreeuwers, M. Schils, D. Maltoni, P. Grother, S. Marcel, et al., Biometric systems under morphing attacks: Assessment of morphing techniques and vulnerability reporting, in: 2017 International Conference of the Biometrics Special Interest Group (BIOSIG), IEEE, 2017, pp. 1–7.

[7] S. Venkatesh, K. Raja, R. Ramachandra, C. Busch, On the influence of ageing on face morph attacks: Vulnerability and detection, in: 2020 IEEE International Joint Conference on Biometrics (IJCB), IEEE, 2020, pp. 1–10.

[8] M. Ferrara, A. Franco, D. Maltoni, C. Busch, Morphing attack potential, in: 2022 International Workshop on Biometrics and Forensics (IWBF), IEEE, 2022, pp. 1–6.

[9] J. M. Singh, R. Ramachandra, Deep composite face image attacks: Generation, vulnerability and detection, IEEE Access.