Multimodal spoofing and adversarial examples countermeasure for speaker verification

by

Karthik Ramesh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Authentication mechanisms have always been prevalent in our society — even as far back as Ancient Mesopotamia in the form of *seals*. Since the advent of the digital age, the need for a good digital authentication technique has soared stemming from the widespread adoption of online platforms and digitized content.

Audio-based authentication like speaker verification has been explored as another mechanism for achieving this goal. Specifically, an audio template belonging to the authorized user is stored with the authentication system. This template is later compared with the current input voice to authenticate the current user.

Audio spoofing refers to attacks used to fool the authentication system to gain access to restricted resources. This has been proven to effectively degrade the performance of a variety of audio-authentication methods. In response to this, spoofing countermeasures for the task of anti-spoofing have been developed that can detect and successfully thwart these types of attacks.

The advent of deep learning techniques and their usage in real-life applications has led to the research and development of various techniques for purposes ranging from exploiting weaknesses in the deep learning model to stealing confidential information. One of the ways in which the deep learning-based audio authentication model can be evaded is the usage of a set of attacks that are known as adversarial attacks. These adversarial attacks consist of adding a carefully crafted perturbation to the input to elicit a wrong inference from the model.

We first explore the performance that multimodality brings to the anti-spoofing task. We aim to *augment* a unimodal spoofing countermeasure with visual information to identify whether it can improve performance. Since visuals can serve as an additional domain of information, we experiment with whether the existing paradigm of using unimodal spoofing countermeasures for anti-spoofing can benefit from this new information. Our results indicate that augmenting an existing unimodal countermeasure with visual information does not provide any performance benefits. Future work can explore more tightly coupled multimodal models that use objectives like contrastive loss.

We then study the vulnerability of deep learning-based multimodal speaker verification to adversarial attacks. In multimodal speaker verification, the vulnerability has not been established and we aim to accomplish this. We find that the multimodal models are heavily reliant on the visual modality and that attacking both modalities lead to a higher attack success rate. Future work can move on to stronger attacks by applying adversarial attacks to bypass the spoofing countermeasure and speaker verification.

Finally, we investigate the feasibility of a generic evasion detector that can block both adversarial and spoofing attacks. Since both the spoofing and adversarial attacks target speaker verification models, we aim to add an adversarial attack detection mechanism — feature squeezing — onto the spoofing countermeasure to achieve this. We find that such a detector is feasible but involves a significant reduction in the identification of genuine samples. Future work can explore combining adversarial training as a defense for attacks that target the complete spoofing countermeasure and speaker verification pipeline.

Acknowledgements

I would like to thank Professor Asokan whose guidance and support made this possible. I would like to thank Compute Canada (Digital Research Alliance of Canada as of April 1, 2022) for providing the compute resources required for the experiments.

Dedication

To my family, whose never-ending love and support made this possible.

To Nivetha, whose patience knows no bounds.

Table of Contents

List of Figures x

List of Tables xiv

List of Abbreviations xvi

1 Introduction 1

2 Background 4

2.1 Speaker Verification 4

2.2 Deep Learning 6

2.2.1 Feed-Forward Neural Networks 7

2.2.2 Convolutional Neural Networks 8

2.2.3 Recurrent Neural Networks 10

2.3 Adversarial Examples 13

2.3.1 Adversarial Example Generation Algorithms 14

2.3.2 Defense Mechanisms 17

2.4 Spoofing 18

2.5 Metrics 19

3 Problem Statement **21**
3.1 System Model . . . 21
3.2 Adversary Model . . . 21
3.2.1 Spoofing Attacks . . . 22
3.2.2 Adversarial Attacks . . . 23
3.3 Research Questions & Metrics . . . 24

4 Multimodal Spoofing Countermeasure **26**
4.1 Dataset . . . 27
4.2 Video Macro Alignment . . . 29
4.3 Methodology . . . 30
4.4 Model Architecture . . . 31
4.4.1 Audio Model . . . 31
4.4.2 Video Model . . . 32
4.4.3 Fusion Model . . . 33
4.4.4 Classification Model . . . 34
4.5 Results & Discussion . . . 34

5 Adversarial Attack on Multimodal Speaker Verification **37**
5.1 Dataset . . . 37
5.2 Automatic Speaker Verification models . . . 38
5.3 Attack Methodology . . . 39
5.4 Results & Discussion . . . 41
5.4.1 Results for adversary model `Read-Only-Template-Access` . . . 42
5.4.2 Results for adversary model `No-Template-Access` . . . 43

6 Malicious Sample Detector **49**
6.1 Dataset . . . 50
6.2 Methodology . . . 50
6.3 Results & Discussion . . . 51

7 Related Work **56**

8 Conclusion & Future Work **59**

References **61**

APPENDICES **69**

A Additional results for Chapter 5 **70**

List of Figures

2.1 Speaker verification pipeline (top) and speaker identification (bottom) pipeline. 5

2.2 Four layer feed-forward neural network 8

2.3 Max pooling operation using a stride of 2 and filter size of 2x2 9

2.4 Inner Workings of an RNN. The input x_t combined with the hidden representation h_{t-1} is used to produce the output h_t. 11

2.5 Inner Workings of an LSTM [38]. There are four gates that control the flow of information, namely, the forget, input, candidate, and output gates. The gates use the previous hidden state h_{t-1} and the current input x_t to control information flow. 12

2.6 Basic pipeline for any model to use the feature squeezing technique. $D()$ is a distance metric. pred_f is the predictions of the squeezed input. 17

3.1 System model depicting the complete flow of data from the capture of input utterance to the spoofing countermeasure and speaker verification models. It also shows the connections the physical access and logical access spoofs target. 22

4.1 Architecture of the proposed multimodal spoofing CounterMeasure (CM). There are four components in the multimodal CM, namely the Audio, Video, Fusion, and Classification models 32

5.1 Depiction of how adversary models compute the perturbation using a multimodal ASV model. For `Read-Only-Template-Access`, we use (A_c, V_c) and (A_t, V_t) to compute the perturbations. For `No-Template-Access`, we use (A_c, V_c) and (A_r, V_r) to compute the perturbations. 40

5.2 Depiction of how the robustness of ASV models is tested. A_c'' and V_c'' are the perturbed audio and visual inputs. A_t and V_t are the templates of the victim y. 41
5.3 Figure showing the white-box attack targeting the audio-visual modality on MV for Read-Only-Template-Access 41
5.4 White-box attack on OPV for Read-Only-Template-Access 45
5.5 Black-box attack targeting OPV where source models are unimodal for Read-Only-Template-Access 46
5.6 Black-box attack targeting OPV where source models are multimodal for Read-Only-Template-Access 47
5.7 Figure showing the white-box attack targeting the audio-visual modality on OPV for No-Template-Access 48
5.8 Figure showing the black-box attack targeting the audio-visual modality by transfer from MV to OPV for No-Template-Access 48

6.1 Feature squeezing pipeline when combined with spoofing countermeasures. 51

A.1 Figure showing the black-box attack targeting the audio modality by transfer from 1D-CNN to MV for Read-Only-Template-Access 70
A.2 Figure showing the black-box attack targeting the audio modality by transfer from 1D-CNN to OPV for Read-Only-Template-Access 71
A.3 Figure showing the white-box attack targeting the audio modality on MV for Read-Only-Template-Access 71
A.4 Figure showing the white-box attack targeting the audio-visual modality on MV for Read-Only-Template-Access 72
A.5 Figure showing the white-box attack targeting the visual modality on MV for Read-Only-Template-Access 72
A.6 Figure showing the black-box attack targeting the audio modality by transfer from MV to OPV for Read-Only-Template-Access 73
A.7 Figure showing the black-box attack targeting the audio-visual modality by transfer from MV to OPV for Read-Only-Template-Access 73
A.8 Figure showing the black-box attack targeting the visual modality by transfer from MV to OPV for Read-Only-Template-Access 74

A.9 Figure showing the black-box attack targeting the audio modality by transfer from OPV to MV for Read-Only-Template-Access 74
A.10 Figure showing the black-box attack targeting the audio-visual modality by transfer from OPV to MV for Read-Only-Template-Access 75
A.11 Figure showing the black-box attack targeting the visual modality by transfer from OPV to MV for Read-Only-Template-Access 75
A.12 Figure showing the white-box attack targeting the audio modality on OPV for Read-Only-Template-Access 76
A.13 Figure showing the white-box attack targeting the audio-visual modality on OPV for Read-Only-Template-Access 76
A.14 Figure showing the white-box attack targeting the visual modality on OPV for Read-Only-Template-Access 77
A.15 Figure showing the black-box attack targeting the audio modality by transfer from X-Vec to MV for Read-Only-Template-Access 77
A.16 Figure showing the black-box attack targeting the audio modality by transfer from X-Vec to OPV for Read-Only-Template-Access 78
A.17 Figure showing the black-box attack targeting the audio modality by transfer from 1D-CNN to MV for No-Template-Access 78
A.18 Figure showing the black-box attack targeting the audio modality by transfer from 1D-CNN to OPV for No-Template-Access 79
A.19 Figure showing the white-box attack targeting the audio modality on MV for No-Template-Access 79
A.20 Figure showing the white-box attack targeting the audio-visual modality on MV for No-Template-Access 80
A.21 Figure showing the white-box attack targeting the visual modality on MV for No-Template-Access 80
A.22 Figure showing the black-box attack targeting the audio modality by transfer from MV to OPV for No-Template-Access 81
A.23 Figure showing the black-box attack targeting the audio-visual modality by transfer from MV to OPV for No-Template-Access 81
A.24 Figure showing the black-box attack targeting the visual modality by transfer from MV to OPV for No-Template-Access 82

A.25 Figure showing the black-box attack targeting the audio modality by transfer from OPV to MV for No-Template-Access 82
A.26 Figure showing the black-box attack targeting the audio-visual modality by transfer from OPV to MV for No-Template-Access 83
A.27 Figure showing the black-box attack targeting the visual modality by transfer from OPV to MV for No-Template-Access 83
A.28 Figure showing the white-box attack targeting the audio modality on OPV for No-Template-Access 84
A.29 Figure showing the white-box attack targeting the audio-visual modality on OPV for No-Template-Access 84
A.30 Figure showing the white-box attack targeting the visual modality on OPV for No-Template-Access 85
A.31 Figure showing the black-box attack targeting the audio modality by transfer from X-Vec to MV for No-Template-Access 85
A.32 Figure showing the black-box attack targeting the audio modality by transfer from X-Vec to OPV for No-Template-Access 86

List of Tables

4.1 Description of the attacks in BTAS. Following the process mentioned in the "Description" column of the Table, the audio is captured by the microphone of the Automatic Speaker Verification (ASV) system. 28

4.2 Architecture of the Video Model 33

4.3 Results of the spoofing countermeasure models on the test set of BTAS. The models above the first horizontal dividing line are unimodal while the rest are multimodal. Each model is trained three times. The Equal Error Rate (EER) results here are the mean±standard-deviation of the EERs of three different models. 34

4.4 Run-Time of the CMs in milliseconds (ms). 36

5.1 Evaluation of the 4 ASV models on the test set of VOXCELEB1. The models above the horizontal dividing line are unimodal while the rest are multimodal. 39

6.1 Adversarial attack success rate (%) on the spoofing CounterMeasure-Automatic Speaker Verification (CM-ASV) pipeline. The success rate is presented as the mean±standard-deviation across the success rates of the four runs. Each attack has a source ASV from which it is generated and a **Modality** it is perturbing. Each defense mechanism consists of a target CM and the attack success rates presented here are the percentage of examples that bypass the target CM and target ASV out of the examples that bypass the target ASV. For this scenario the target ASV is the OPV. We use different colors to indicate the best (lowest) and worst (highest) metrics (based on mean) for each epsilon value. We indicate all models without defense using underline. . . 53

6.2 Adversarial attack success rate (%) on the CM-ASV pipeline. The success rate is presented as the mean±standard-deviation across the success rates of the four runs. Each attack has a source ASV from which it is generated and a **Modality** it is perturbing. Each defense mechanism consists of a target CM and the attack success rates presented here are the percentage of examples that bypass the target CM and target ASV out of the examples that bypass the target ASV. For this scenario the target ASV is the MV. We use different colors to indicate the best (lowest) and worst (highest) metrics (based on mean) for each epsilon value. We indicate all models without defense using underline. 54

6.3 Table showing the True Positive Rate (TPR) (%) of the CMs from Tables 6.1 and 6.2 on the test set of BTAS and VOXCELEB1. We use different colors to indicate the best (highest) and worst (lowest) metrics (based on mean) for each epsilon value. We indicate all models without defense using underline. 55

List of Abbreviations

ASV Automatic Speaker Verification xiv, xv, 1, 2, 14, 15, 18, 19, 21–25, 27, 28, 37–40, 44, 49–54, 57–59

CM spoofing CounterMeasure x, xiv, xv, 1, 18, 19, 21, 22, 24, 26, 27, 30–32, 35, 36, 49–55, 57–60

CM-ASV spoofing CounterMeasure-Automatic Speaker Verification xiv, xv, 21, 30, 51–54, 59

CNN Convolutional Neural Network 8, 10, 56

DNN Deep Neural Network 56

EER Equal Error Rate xiv, 19, 24, 26, 34, 38, 39, 50, 53

FAR False Acceptance Rate 19, 54

FFN Feed-Forward Neural Network 7–9, 34

FGSM Fast Gradient Sign Method 15, 42, 43

FN False Negative 19, 20

FNR False Negative Rate 19

FP False Positive 19, 20

FPR False Positive Rate 19

FRR False Rejection Rate 19, 54

GMM Gaussian Mixture Model 56

LPS Log Power Spectrum 31

LSTM Long Short-Term Memory 11, 31, 33, 34

MIFGSM Momentum Iterative Fast Gradient Sign Method 16, 42

NLP Natural Language Processing 2

PGD Projected Gradient Descent 16, 42

RNN Recurrent Neural Network 10, 11

TN True Negative 19, 20

TP True Positive 19, 20

TPR True Positive Rate xv, 25, 49, 50, 53–55

UBM Universal Background Model 56

Chapter 1

Introduction

The sound produced by each individual is unique due to varying physical differences in their vocal organs. In addition, each individual has their unique characteristics of speaking such as accent and vocabulary thereby leading to more varieties in their sound [24]. These differences make it feasible to identify a person given a sample of their speech. *Automatic speaker recognition* refers to tasks associated with identifying the speaker given a voice sample belonging to that speaker [24]. The task of *Automatic Speaker Verification (ASV)* [4] is to determine if two input speech samples belong to the same speaker. ASV is one of the ways by which automatic speaker recognition can be accomplished — it is sometimes colloquially known by the name of its superset, that is, automatic speaker recognition. This has a wide variety of use-cases ranging from controlling access to secure vaults, bank accounts, and personal smart devices to aiding forensic investigators in suspect recognition.

The research community has found that current ASV mechanisms are susceptible to *malicious actors/adversaries* through a set of attacks called *spoofing attacks* [60]. The spoofing attacks are used by adversaries seeking to bypass the protection that ASV offers by creating *spoofs*. These spoofs are generated to mimic an input that the system recognizes as an authorized user's input. But in actuality, it is an input that the adversary has crafted using spoofing attacks to bypass the ASV system. These spoofs are typically generated using *replayed audio*, *voice conversion*, and *speech synthesis* mechanisms. The mechanisms that were proposed to defend against these spoofs are known as *spoofing CounterMeasures (CMs)* and the task of identifying and blocking these spoofs is called *anti-spoofing* [60]. A CM typically consists of another system separate from the ASV whose purpose is to detect and block the inputs that are spoofs. The AVspoof [15] dataset was one of the first standardized datasets available in the domain to research CMs in a generalized manner.

Challenges such as the BTAS [26] and ASVspoof [60] soon became available that helped promote interest in the research.

Deep learning models have shown significant performance in the computer vision domain for unimodal tasks such as face recognition [46], object detection [42], and for multimodality tasks like image captioning [57]. The Natural Language Processing (NLP) research community has also developed a significant number of models and breakthroughs that showcase improvements in tasks like machine translation [56], language modeling [40], and representation learning [35]. Naturally, deep learning models have also been adopted for usage in the audio domain for a variety of tasks such as speech separation [14], speech synthesis [61], speech recognition [2], and speaker recognition [50].

With the evolution of deep learning technologies, the research community has also become invested in investigating the vulnerability of these new technologies to malicious actors. One of the ways in which the deep learning models, in general, are vulnerable is to a set of attacks known as *adversarial attacks* [51]. Adversarial attacks involve crafting an *example* specifically to mislead the deep learning models, that is, these examples are crafted by adding a small perturbation to it which results in the example being incorrectly classified (in the case of a classification model) by the models. *Adversarial examples* have also been shown to transfer well [19] between models and datasets, thereby becoming an avenue of threat that needs to be addressed. Various defense mechanisms have also been developed by the research community to thwart these malicious actors ranging from *adversarial training* [19] to *input defense* [70] mechanisms.

This thesis will focus on the intersection of these three aspects in the deep learning domain: speaker verification, spoofing countermeasures, and adversarial attacks.

We investigate the performance that multimodality brings to the table in the anti-spoofing task. We investigate the vulnerability of the audio-visual speaker verification systems to adversarial attacks. We wrap up this work with the investigation of the feasibility of a malicious sample detector by having it detect both adversarial examples and spoofs that target ASVs.

The contributions of this thesis are listed below:

- We evaluate the effectiveness of adding visual information to an audio-only spoofing countermeasure. This countermeasure jointly uses visual information along with spoofed audio information to decide on whether the given audio input is a spoof. This is compared with a state-of-the-art unimodal spoofing countermeasure to identify the performance that multimodality offers — Section 4. We show that when directly augmenting an existing unimodal spoofing countermeasure with visual information, the multimodal model does not produce any performance improvements.

- We reproduce two unimodal and two multimodal speaker verification systems. We study the damage that adversarial attacks can cause to multimodal systems. We attack the systems using three different adversarial attacks. We also consider two attack scenarios and demonstrate their vulnerability under them — Section 5. We show that under the more difficult adversary model, attacks succeed in less than 10% of the cases.

- We study the robustness of spoofing countermeasures to the adversarial examples generated for speaker verification systems. We propose a mechanism to defend against spoofs and adversarial examples simultaneously using a unified detector — Section 6. This mechanism is feasible but comes at a significant reduction in legitimate examples detection rates.

Chapter 2

Background

2.1 Speaker Verification

The task of *speaker verification* is often misinterpreted as *speaker identification* while they are two different concepts. Given two input *utterances*, the goal of speaker verification is to declare whether they both have been uttered by the same speaker or not. The task of speaker identification on the other hand is to determine the speaker of a given input utterance.

When we talk of speaker identification in this thesis, we are referring to the version of the problem where the speakers are limited to a specific set and it does not support the enrollment of new speakers at a later stage without a new model [4]. On the other hand, the speaker verification system does not have any restrictions regarding the identity of the speakers, that is, the speakers need not be known in advance. The formulations for the speaker verification and identification are depicted below:

$$verified = F_{verification}(utt_X, utt_Y, W_{verification}) \geq \tau_{verification} \tag{2.1}$$

$$identity = \arg\max_{u \in U} F_{identification}(utt_X, W_{identification})[u] \geq \tau_{identification} \tag{2.2}$$

Equation 2.1 depicts the formulation of the speaker verification model while Equation 2.2 depicts the formulation of the speaker identification model. Figure 2.1 visualizes the differences between the two tasks. F corresponds to the speaker verification/identification model and W is the set of parameters associated with the verification/identification model. utt_X and utt_Y are the input utterances. U corresponds to the global set of speakers that the speaker identification model is trained to identify. $\tau_{verification}$ is the *threshold* used by the

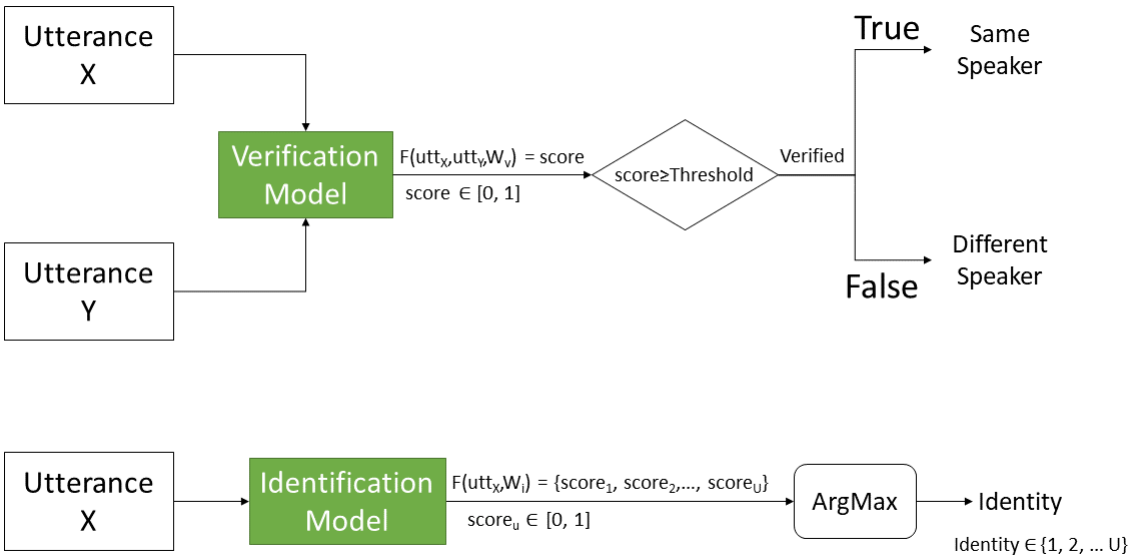


Figure 2.1: Speaker verification pipeline (top) and speaker identification (bottom) pipeline.

speaker verification model to distinguish whether the input utterance pair have come from the same speaker or not. $\tau_{identification}$ is the *threshold* used by the speaker identification model to determine whether the given input utterance is sufficiently distinguishable as belonging to a speaker.

The task of speaker verification can be accomplished with either embedding-similarity based models [50] or end-to-end based models [48]. The embedding-similarity models can be separated into a front-end feature (*embedding*) extractor and a back-end *similarity score* computation. The front-end works on the input utterance and outputs a high-dimensional feature vector. This front-end is typically directly trained on some objective with just a single input utterance rather than a pair of utterances. The back-end takes a pair of these high-dimensional inputs and outputs a similarity score. This similarity score, if above a threshold, leads to the conclusion that these inputs are similar and belong to the same speaker. On the other hand, end-to-end models directly generate the similarity score between two given input utterances during training and are trained to optimize their parameters on that score. During inference, a threshold function similar to embedding-similarity models is used to determine speaker similarity.

2.2 Deep Learning

First, we explain certain notations and terms used in the remainder of the thesis. Then we go on to explain certain deep learning mechanisms that will be used in the remainder of the thesis.

A supervised deep learning model can be characterized as a function $F_\theta()$ where θ corresponds to the *parameters* of the model. The *dataset* used by the model can be depicted as the set $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ where x_i is an input data, y_i is the corresponding output, and N is the number of *samples* in the dataset. *Hyperparameters* of a model are parameters that control the learning process and are set prior to the learning process.

For audio-based classification tasks like anti-spoofing and speaker identification, the input $x_i \in \mathbb{R}^{t \times d}$ where t is the number of timesteps and d is the dimensionality of the input — the input is typically an *utterance* or a feature derived from that utterance. The output y_i takes a value from the set $\{1, 2, ...C\}$ where C is the number of classes that the input can belong to in the given problem. The output y can be called as the *class*/*label* and the function F as the *classifier*/*model*.

A dataset is typically split into *training* and *testing* sets which are used for training and testing the model respectively. Additionally there can be a *validation* (*development*) set used to tune the hyperparameters of the model.

Models consist of different behaviors depending on whether they are in the process of training/testing. The training phase is where the model is taught to perform a certain task with the help of a *loss function*. The loss function acts as a proxy for estimating the performance of the model (with its current set of parameters) for a given task. Combining this with an optimization function (*optimizer*), the model's parameters are updated to perform better on the same task. During the testing phase, the model is evaluated on a separate split of the dataset to indicate the performance of the model.

The training process of a model consists of a *forward pass* and a *backward pass*. The forward pass takes a given input and *passes* it through the layers of a model thereby generating the output and loss — loss is generated using the loss function. On the other hand, the backward pass consists of computing the gradients of the loss with respect to the various parameters of a model — like weights and biases — using the *backpropagation* algorithm and *passing* it backward — going from the last parameter to the first — through a model. These gradients are used by an optimizer to update the parameters of a model.

Equation 2.3 shows the formulation of a loss function for a model and Equation 2.4 shows the update process for the parameters. J is the loss function, $p \in [1, P]$ where P is

the number of parameters to be updated, and α is the learning-rate.

$$J(x, \theta, y) = \begin{cases} -log(F_\theta(x)) & y = 1 \\ -log(1 - F_\theta(x)) & y = 0 \end{cases} \tag{2.3}$$

$$\theta_p = \theta_p - \alpha . \frac{\partial}{\partial \theta_p} J(x, \theta, y) \tag{2.4}$$

2.2.1 Feed-Forward Neural Networks

The Feed-Forward Neural Network (FFN) [17] is one of the most widely used deep-learning architectures in the current state-of-the-art models. It is also one of the basic building blocks for solving complex problems using neural network architectures.

FFNs can be thought of as a more complex version of the standard logistic regression model [17]. The basic binary logistic regression model has parameters θ, takes in an input x and predicts the probability $\mathbb{P}(pred = 1|x, \theta)$ and its loss function is described in Equation 2.3. The architecture of the logistic regression model can be described as follows:

$$h(x) = w^T x + b \tag{2.5}$$

$$F(x) = \sigma(h(x)) \tag{2.6}$$

Equation 2.5 shows how the intermediate values of the model are computed. $w \in \mathbb{R}^{d \times 1}$ is the *weight* parameter of the model, $b \in \mathbb{R}^1$ is the *bias* parameter of the model and $x \in \mathbb{R}^d$ is the input feature vector to the model. After computation of the intermediate values, a sigmoid/logistic activation function (σ) is used to bound the values (Equation 2.6) following which the loss is computed and parameters of the model are updated. This results in logistic regression modeling a linear boundary between classes and is unable to effectively model non-linearity.

FFNs can model non-linear boundaries using a non-linear activation function at their hidden layer. Each layer of the FFN has a set of *neurons* and we can consider a single layer FFN as a non-linear form of the logistic regression model. In FFNs, each layer consists of a set of neurons and the network consists of a set of layers. Figure 2.2 visually represents a four layer FFN — three hidden layers and one output layer — where each hidden layer has a non-linear activation (A_n) function.

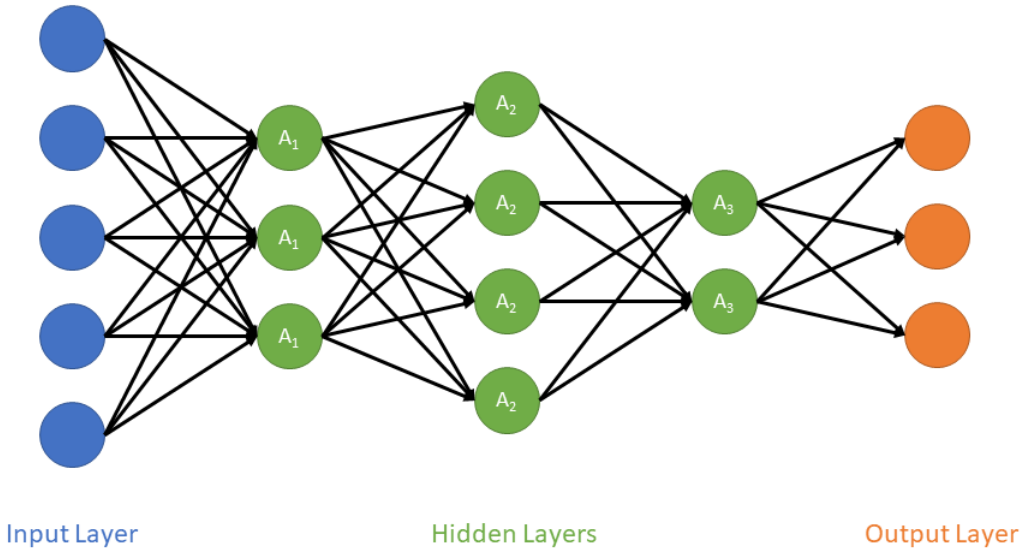


Figure 2.2: Four layer feed-forward neural network

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [29] are one of the foundational building blocks of modern computer vision. They have been used in various tasks like object detection [42] and face recognition [46]. Here, we will discuss the 2D CNNs for classification.

A CNN can be designed to take input consisting of raw pixel values of images of shape $\mathbb{R}^{w \times h \times c}$ where w, h, and c are the width, height, and number of channels of the image respectively. To use it for a sequence of images like in a video there are multiple techniques to process them including using 2D CNNs with grayscale images [44] where the channel dimension represents timesteps or using 3-D CNNs [54].

A neural network based on CNN is mainly composed of three components: convolution layers, pooling layers, and classifiers.

A convolution layer consists of an affine transformation followed by an activation function similar to the FFN. The convolution operation is carried out by a set of filters of shape $\mathbb{R}^{x \times y \times c}$ where x, y, and c are the width, height, and channel dimensions of the filter respectively. Each of these filters is passed over the feature map from the output of the previous layer or the input. These filters perform an element-wise multiplication followed by a bias summation.

It can be represented by Equations 2.7 and 2.8 where i, j refer to the spatial coordinates on the input feature map and i', j' refer to the spatial coordinates on the output feature map. k is the current filter number, I is the input feature map from the previous layer, F is the filter containing weights, and ϕ is the activation function.

$$h^k_{i',j'} = \sum_{x'=0}^{x-1} \sum_{y'=0}^{y-1} \sum_{c'=0}^{c-1} I_{i+x',\ j+y',\ c'} \cdot F^k_{x',\ y',\ c'} + b^k \tag{2.7}$$

$$C^k_{i',j'} = \phi(h^k_{i',j'}) \tag{2.8}$$

The pooling layers are used to aggregate the information into a more compact form. Max pooling is depicted in Figure 2.3 which is one of the most widely used types of pooling. Here, a window is moved over the input feature map similar to the way it is done in convolution. The difference is that from within each window, the maximum value is taken as output. The output feature map is typically smaller than the input due to a high stride value during the pooling operation.

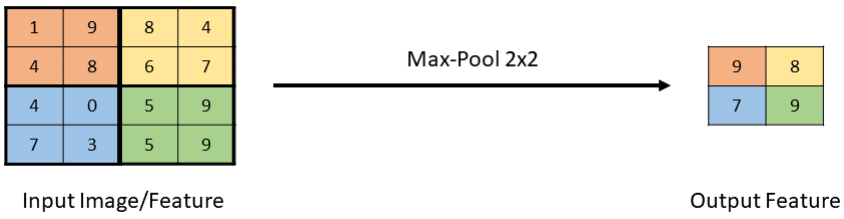


Figure 2.3: Max pooling operation using a stride of 2 and filter size of 2x2

The classifier typically consists of an FFN whose input is the feature map of the convolution and pooling layers that is flattened into a vector. This is then combined with a loss function and optimizer to train the model.

Depthwise Separable Convolutions: Depthwise separable convolutions were introduced in [21] to provide a light-weight alternative to the standard convolution operation. The idea behind separable convolutions is to replace the standard convolution operation with two new operations, namely, depthwise convolution and pointwise convolution. Depthwise convolution consists of using a new filter for every channel in the input feature map. Depthwise convolution results in parameter size (ignoring bias) of $R^{x \times y \times 1 \times f}$ where x, y, and f are the width, height, and number of filters respectively. Pointwise convolution

consists of one standard convolution but with a filter whose spatial dimensions are 1x1. Pointwise convolution results in parameter size (ignoring bias) of $R^{1\times1\times c\times f}$ where c and f are the number of channels and filters respectively.

Light CNN: The Light CNN [28] is one of the architectures proposed for the anti-spoofing task and has become a baseline model for evaluating the current spoofing attacks. A key characteristic of this model is the Max-Feature-Map (MFM) activation. This activation is based upon the usage of a maxout function which they've identified as being useful for the anti-spoofing task.

Attention: While there is no one concrete definition for attention, it can be broadly thought of as a mechanism that is used by a neural network to weigh features based on their importance to a task. This allows the network to select the most important features corresponding to a task for a given input and facilitates in achieving higher performing and interpretable models. The architecture by [69] is one such example where an attention mechanism is used to improve the performance of the image captioning model. Image captioning models typically use an encoder-decoder architecture where the encoder compresses information from the visual input into a single vector and the decoder uses this compressed information as context to generate the output captions. Xu et al. [69] proposed to let the encoder produce a set of vectors and they add an attention component to the decoder thereby letting the decoder choose the important information from the output of the encoder at every timestep.

2.2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a set of networks that are highly effective at dealing with sequential data. They have been used heavily in the text and audio domains for tasks such as language modeling [34; 40], machine translation [3], speech enhancement [37] and automatic speech recognition [20].

A standard RNN can be seen in Figure 2.4 and is formulated as Equations 2.9 and 2.10. The trainable weight parameters W_{hh} and W_{xh} transform the hidden state of the previous timestep and the input of the current timestep respectively — they are transformed into the hidden representation of the current timestep. W_{ho} transforms the current hidden state to output class probabilities — for tasks like language modeling. b_h and b_o are trainable bias parameters. They work on an input sequence of features $x_1, x_2, ..., x_t$ to produce an output

sequence of class probabilities $o_1, o_2, ..., o_t$ from an intermediate sequence of representations $h_1, h_2, ..., h_t$.

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{2.9}$$
$$o_t = softmax(W_{ho}h_t + b_o) \tag{2.10}$$

One drawback of the standard RNN is their poor performance in long input sequences —

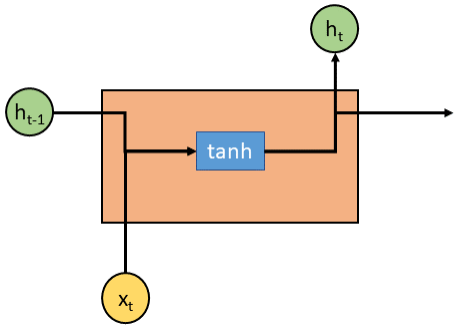


Figure 2.4: Inner Workings of an RNN. The input x_t combined with the hidden representation h_{t-1} is used to produce the output h_t.

they have difficulty capturing long-term dependencies due to exploding/vanishing gradients [39]. The Long Short-Term Memory (LSTM) (Figure Figure 2.5) is one of the variants of the RNN proposed to tackle that difficulty. It can be formulated as follows:

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f) \tag{2.11}$$
$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t + b_i) \tag{2.12}$$
$$o'_t = \sigma(W_{ho'}h_{t-1} + W_{xo'}x_t + b_{o'}) \tag{2.13}$$
$$c'_t = tanh(W_{hc'}h_{t-1} + W_{xc'}x_t + b_{c'}) \tag{2.14}$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t \tag{2.15}$$
$$h_t = o'_t * tanh(c_t) \tag{2.16}$$
$$o_t = h_t \tag{2.17}$$

The LSTM's inner workings can be seen in Figure 2.5. The LSTM has a cell state which is used to pass information from one timestep to the next while also allowing the gradients to flow back without vanishing/exploding. It uses gates to control the flow of information through the cell state thereby allowing it to keep useful information for future timesteps to access. It has a forget gate (f_t), input gate (i_t), and candidate gate (c'_t) to control changes

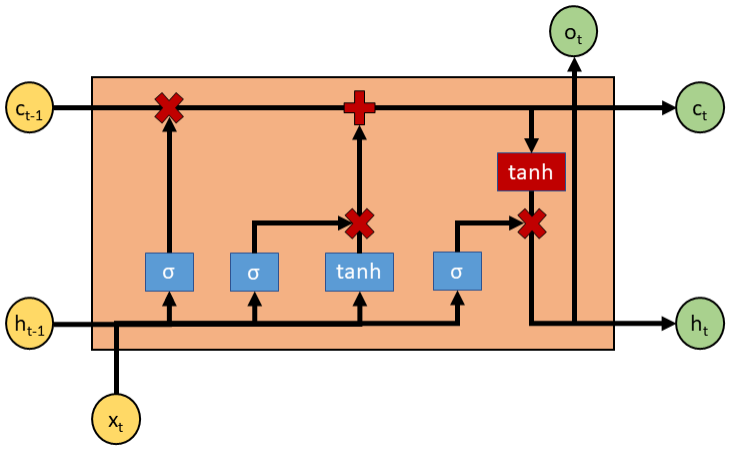


Figure 2.5: Inner Workings of an LSTM [38]. There are four gates that control the flow of information, namely, the forget, input, candidate, and output gates. The gates use the previous hidden state h_{t-1} and the current input x_t to control information flow.

to cell state and output gate (o_t') to prepare data for the next timestep. The forget gate is used to retire old information. The input and candidate gates are used to control what is newly added from the current timestep to the cell state. The output gate is used to control the output of the current timestep as well as the hidden state passed to the next timestep.

2.3 Adversarial Examples

Although adversarial examples exist for mechanisms other than deep learning models [5], this thesis's focus is on deep learning models and therefore we talk about them in this section.

Adversarial examples are *data instances* created by adding a carefully selected small *perturbation* which forces the deep learning model to misclassify (in the case of a classification model) the data instance. One key motivation as to why these data instances are worthy of being investigated is that they are not distinguishable by the human eye (in the case of images) — they appear visibly the same as the non-perturbed (clean) data instance.

The perturbation can be crafted to attack the model in an untargeted or targeted manner. An untargeted attack aims to get the model to misclassify a data instance as *any class other than the true class*. More formally, in the case of a classification scenario, given a model $F()$, a data instance x, classes {1, 2, ...C}, perturbation ϵ, an untargeted attack can be formulated as follows:

$$\max L(F(x+\epsilon),\ c) \quad s.t.\ F(x+\epsilon) \neq c,\ d(x,\ x+\epsilon) \leq \eta \tag{2.18}$$

$$\min d(x,\ x+\epsilon) \quad s.t.\ F(x+\epsilon) \neq c \tag{2.19}$$

In the above equations, $L()$ refers to the loss function used as a metric to optimize the model for the associated task, $d()$ is the metric used to measure the distance between the perturbed and unperturbed data instances and c is the class of the unperturbed data instance. η is the constraint on the distance between perturbed and clean data instances. Equation 2.18 optimizes a loss function to achieve for an incorrect classification while Equation 2.19 optimizes a distance function to obtain a minimally perturbed data instance that is misclassified. Either of them can be used to achieve an untargeted attack.

The aim of a targeted attack is to get the model to misclassify the data instance as an instance of a specific class which can be formulated as:

$$\min L(F(x+\epsilon),\ c') \quad s.t.\ F(x+\epsilon) = c',\ d(x,\ x+\epsilon) \leq \eta \tag{2.20}$$

$$\min d(x,\ x+\epsilon) \quad s.t.\ F(x+\epsilon) = c' \tag{2.21}$$

In the above equations, c' is the class that we want the perturbed data instance to be classified as. Similar to Equations 2.18 and 2.19, Equations 2.20 and 2.21 optimize for the loss and distance function respectively.

The L_p metric is usually used as the distance metric and is formulated as follows:

$$L_p(x, \bar{x}) = (\sum_{i=1}^{N} |x_i - \bar{x}_i|^p)^{1/p}$$

x and $\bar{x}$ are the clean and perturbed data instances respectively. N is the number of dimensions. The L_p metrics used in attacks are typically L_0, L_2, and L_∞. The L_0 metric ensures that the number of points perturbed in a data instance is within a limit. The L_2 metric is the Euclidean distance and ensures that x and $\bar{x}$ are within a given Euclidean distance from each other. The L_∞ ensures that the maximum perturbation on any data point within a data instance is bounded.

Attacks can be conducted in two ways: single-step or multi-step. In single-step, the perturbed data instance has to be generated in one pass, that is. the attacker cannot do multiple forward and backward passes to generate the perturbation to be added. In multi-step, there can be multiple iterations of forward and backward passes to achieve the best-perturbed data instance.

There are multiple conditions under which the robustness of a *target model* can be tested, but the most commonly used ones are the *white-box* and *black-box* scenarios. In the white-box scenario, it is assumed that the attacker has access to everything about the target model like weights, hyperparameters, training datasets, and outputs. This represents the worst-case scenario for the model/defender. The black-box scenario is where the attacker does not have access to the interior workings of the target model but has access to the model's inference. One way this is adopted for the adversarial attack scenario is by assuming the attacker has access to a system similar to the target model — an adversarial example is generated for this new model which is then used to attack the target model. This portrays a more real-world attack — this is also known as *transfer attacks* — that can be conducted by the attacker.

2.3.1 Adversarial Example Generation Algorithms

Here, we aim to elaborate on the adversarial example generation algorithms that we use in the remainder of the thesis. The algorithms are described for multimodal Automatic

Speaker Verification (ASV) models, but the unimodal version can be directly extrapolated.

Algorithm 1: FGSM Algorithm

Input: First speaker data A_x, V_x, Second speaker data A_y, V_y,
Adversarial perturbation ϵ,
Feature Extractor F for the ASV, Model ASV

Output: Perturbed data A''_x, V''_x

$Feats_x = F(A_x, V_x)$
$Feats_y = F(A_y, V_y)$
$Grads = \nabla_{A_x, V_x} ASV(Feats_x, Feats_y)$
$A''_x = A_x + \epsilon \times sign(Grads_{A_x})$
$V''_x = V_x + \epsilon \times sign(Grads_{V_x})$
return A''_x, V''_x

Fast Gradient Sign Method (FGSM): FGSM [19] (Algorithm 1) is one of the first algorithms developed for generating adversarial examples efficiently. In Algorithm 1, ∇_x corresponds to the partial differentiation operator used to find the gradients with respect to x. ASV outputs the similarity of utterances from two speakers — we use cosine distance as the similarity metric in the ASV models. $sign(x)$ is a function used to extract the sign of the real number x. ϵ is the parameter that controls how much the data is perturbed.

Algorithm 2: PGD Algorithm

Input: First speaker data A_x, V_x, Second speaker data A_y, V_y,
Maximum adversarial perturbation ϵ, Step size α, Steps T,
Feature Extractor F for the ASV, Model ASV

Output: Perturbed data A''_x, V''_x

Initialize:
    $A''_x = A_x + \mathcal{U}(-\epsilon, \epsilon)$
    $V''_x = V_x + \mathcal{U}(-\epsilon, \epsilon)$
for $step \leftarrow 1$ **to** T **do**
    $Feats_x = F(A''_x, V''_x)$
    $Feats_y = F(A_y, V_y)$
    $Grads = \nabla_{A''_x, V''_x} ASV(Feats_x, Feats_y)$
    $A''_x = Clip_{A_x, \epsilon}(A''_x + \alpha \times sign(Grads_{A''_x}))$
    $V''_x = Clip_{V_x, \epsilon}(V''_x + \alpha \times sign(Grads_{V''_x}))$
return A''_x, V''_x

Projected Gradient Descent (PGD): PGD [33] is an algorithm (Algorithm 2) that takes a multi-step approach to generating the perturbed data by iteratively perturbing the input to maximize the loss. In Algorithm 2, $Clip_{x,\epsilon}(x'')$ refers to the function used to bound the iteratively perturbed data x'' within an ϵ region of unperturbed data x using the L_∞ metric. The relationship between steps T, step size α, and maximum perturbation ϵ can be written as $\epsilon = T \cdot \alpha$. The "Initialize" section is used to give a uniform random ($\mathcal{U}$) perturbation to the data. The rest of the notations are similar to Algorithm 1.

Algorithm 3: MIFGSM Algorithm

Input: First Speaker data A_x, V_x, Second Speaker data A_y, V_y, Adversarial Perturbation ϵ, Step size α, Steps T, Decay $decay$, Feature Extractor F for the ASV, Model ASV

Output: Perturbed data A''_x, V''_x

Initialize:
- $A''_x = A_x$
- $V''_x = V_x$
- $Momentum_{A''_x} = zeros\_like(A''_x)$
- $Momentum_{V''_x} = zeros\_like(V''_x)$

for $step \leftarrow 1$ **to** T **do**
- $Feats_x = F(A''_x, V''_x)$
- $Feats_y = F(A_y, V_y)$
- $Grads = \nabla_{A''_x, V''_x} ASV(Feats_x, Feats_y)$
- $Grads_{A''_x} = Momentum_{A''_x} = Momentum_{A''_x} \times decay + normalize(Grads_{A''_x})$
- $Grads_{V''_x} = Momentum_{V''_x} = Momentum_{V''_x} \times decay + normalize(Grads_{V''_x})$
- $A''_x = Clip_{A_x,\epsilon}(A''_x + \alpha \times sign(Grads_{A''_x}))$
- $V''_x = Clip_{V_x,\epsilon}(V''_x + \alpha \times sign(Grads_{V''_x}))$

return A''_x, V''_x

Momentum Iterative Fast Gradient Sign Method (MIFGSM): MIFGSM [13] is an algorithm (Algorithm 3) that adds a momentum factor which accumulates the gradients across multiple iterations to obtain a stable perturbation update in successive iterations. In Algorithm 3, *decay* is the parameter used to accumulate the gradients from the prior iterations. *normalize*() function uses the L_1 distance to normalize the gradient information. The "Initialize" uses the $zeros\_like(x)$ function to create an array of zeros of shape of x to initialize the momentum data. The rest of the notations are similar to Algorithms 1 and 2.

2.3.2 Defense Mechanisms

Adversarial Training: One of the most frequently used and widely effective techniques to make a model robust to adversarial attacks is adversarial training [19]. This involves generating adversarial examples and augmenting the training process of the model with this data. This enables the model to be intrinsically robust without any changes in the architecture or pipeline. But, the model does have to be re-trained/finetuned on the augmented data.

Feature Squeezing: Feature squeezing [70] is another method that does not require any change to the model architecture, but, requires a change in the pipeline. It serves as an adversarial example detection method rather than an intrinsic defense method like adversarial training. Here, they "squeeze" the input using a squeezing method and compare the difference in model predictions between the squeezed and un-squeezed inputs to detect whether the input is an adversarial example. Figure 2.6 depicts how this is accomplished.

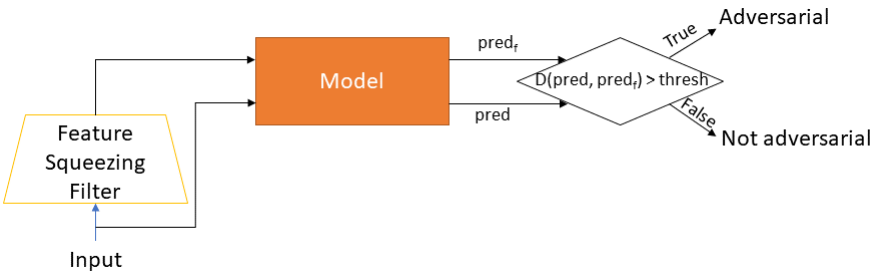


Figure 2.6: Basic pipeline for any model to use the feature squeezing technique. $D()$ is a distance metric. pred_f is the predictions of the squeezed input.

A filter function is used to squeeze the input and the prediction difference between the original input and this new filtered input is compared using a distance metric $D()$. We follow the original work's metric by using the L_1 norm on the probability vector. We use a local smoothing filter as described in [70]. The filter uses runs a sliding window (filter) over the input and applies a blur function like median across the values inside the window.

2.4 Spoofing

Spoofing has been proven to work as a technique to bypass the protection that the ASV offers [60]. The spoofing techniques mainly fall into the following categories: impersonation, replay, speech synthesis, and voice conversion. The impersonation spoof consists of one speaker altering their voice to sound like the *target speaker* — the voice characteristics modified are typically high-level information like prosody whose effects on the performance of the ASV models are often ineffective [15; 11]. As a result, modern spoofing CounterMeasures (CMs) often are not evaluated for this technique.

Replay spoofs consist of recording the audio of a target speaker and playing it back using a set of speakers into ASV system. Speech synthesis uses advanced algorithms and machine learning models to generate an audio recording in the voice of the target speaker. Voice conversion techniques are used to convert the voice of a speaker in a given audio to the voice of the target speaker.

The development of CMs has been prompted due to these techniques. One of the first public datasets that consist of all the three techniques used above is the AVspoof dataset [15]. Challenges such as the ASVspoof series [68; 25; 60] have further been introduced that improved the state-of-the-art of the spoofing CMs along with better spoofs. The ASVspoof 2015 challenge tackles the speech synthesis and voice conversion spoof techniques and focuses on CMs for them. The ASVspoof 2017 challenge focuses on studying CMs for replay attacks detection. The ASVspoof 2019 combines both of the above scenarios and focuses on CMs for all three major spoofing techniques.

In addition, there are two scenarios under which the spoofing attack can be conducted: physical access and logical access. The physical access scenario is where a given spoof is played through a replay device (speaker) and obtained by the capture device (microphone/sensor) of the ASV system before being passed to the ASV — the spoof is subject to the capture device which introduces variability in terms of the environment such as distance from replay device to capture device, the room's acoustic properties, and the replay and capturing devices' properties. The logical access scenario is where the spoof is directly passed to the ASV thereby bypassing the sensor and directly introducing it to the channel between the capture device and ASV.

2.5 Metrics

Here, we will elaborate on the metrics used in the remainder of the thesis. The metrics below are mentioned from the perspective of a binary classification scenario — the two classes are *positive* and *negative*. For ASV models, the positive class corresponds to the inputs originating from the same speaker. For CM models, the positive class corresponds to the inputs that are not spoofs.

True Positive (TP): The number of samples that are *correctly* classified as belonging to the positive class.

True Negative (TN): The number of samples that are *correctly* classified as belonging to the negative class.

False Positive (FP): The number of samples that are *incorrectly* classified as belonging to the positive class. The instances of this error (false positives are an error type) are also known as *false acceptance*.

False Negative (FN): The number of samples that are *incorrectly* classified as belonging to the negative class. The instances of this error (false negatives are an error type) are also known as *false rejection*.

Equal Error Rate (EER): The is the error when the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) are equal.

$$FalsePositiveRate(FPR) = FAR = \frac{FP}{FP + TN} \tag{2.22}$$

$$FalseNegativeRate(FNR) = FRR = \frac{FN}{FN + TP} \tag{2.23}$$

Precision: Out of the samples classified as belonging to the positive class, the fraction that were correctly classified.

$$Precision = \frac{TP}{TP + FP} \tag{2.24}$$

Recall: The fraction of positive samples that have been correctly classified.

$$Recall = \frac{TP}{TP + FN} \tag{2.25}$$

F_1**-Score**: The commonly used F_1-Score is the harmonic mean of the precision and recall.

$$F_1\text{-}Score = 2 \times \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.26}$$

Attack Success Rate: It measures the percentage of samples that are incorrectly classified by an adversarial attack. The below equation assumes that all the inputs are adversarial inputs.

$$AttackSuccessRate = \frac{FP + FN}{TP + TN + FP + FN} \times 100 \tag{2.27}$$

Chapter 3

Problem Statement

3.1 System Model

In real-world deployments, a spoofing CounterMeasure (CM) and an Automatic Speaker Verification (ASV) model are deployed simultaneously and they work in tandem. Figure 3.1 depicts the *spoofing CounterMeasure-Automatic Speaker Verification (CM-ASV)* pipeline. The critical components required for the functioning of the pipeline are an ASV and CM model. An ASV determines whether a given pair of inputs correspond to the same speaker or not, whereas a CM determines whether a given input is a spoof or not. So any input to the ASV in the CM-ASV first has to pass the validation by the CM model that the input is not a spoof.

3.2 Adversary Model

We now define the adversary models when *targeting an ASV* by identifying the capabilities of the adversary for different types of attacks. We consider two types of attacks, namely, spoofing attacks and adversarial attacks.

Figure 3.1 depicts a *control boundary* (green box in the Figure) indicating the parts of CM-ASV that are not accessible by the adversary in the adversary model. There are three different scenarios corresponding to the control boundary:

- With the control boundary as shown in the Figure, the adversary has *physical access* to the capture device and *logical access* to the channel between the capture device

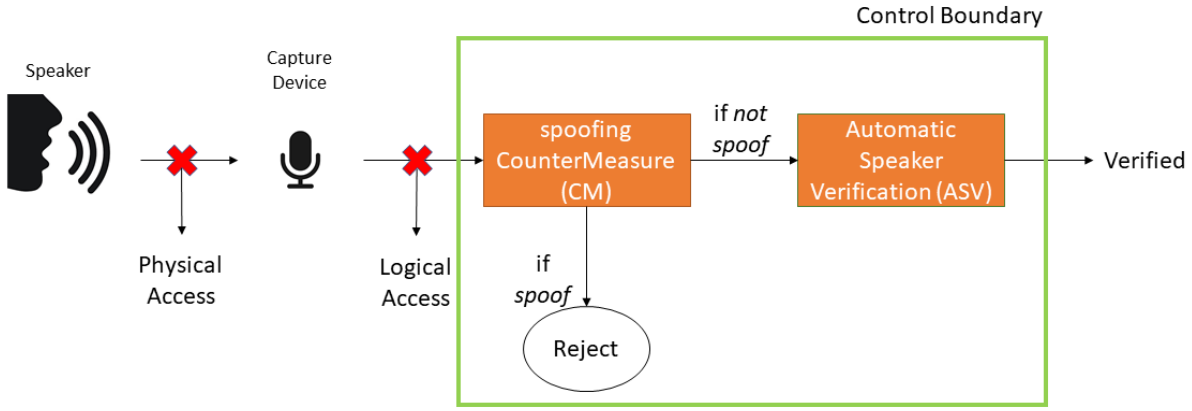


Figure 3.1: System model depicting the complete flow of data from the capture of input utterance to the spoofing countermeasure and speaker verification models. It also shows the connections the physical access and logical access spoofs target.

and the backend (CM/ASV). The adversary can choose to conduct either a physical access attack at the capture device or a logical access attack at the channel.

- The control boundary can extend to include the capture device as well. In this case, the adversary is limited to conducting a physical access attack at the capture device.
- The adversary *may* have knowledge of select parts inside the control boundary thereby making adversarial attacks feasible. This will further explained in Section 3.2.2.

3.2.1 Spoofing Attacks

We now elucidate the types of spoofing scenarios and spoofing attacks the adversary can use against an ASV model. The adversary can craft spoofs under the physical access and logical access scenarios. Figure 3.1 depicts the connections that the physical access spoof and logical access spoof target to create successful spoofing attacks.

The goal of the adversary in spoofing attacks is to gain access to the resource being protected by an ASV model. The CM exists as a mechanism to block this illegal access. So, the spoofs are created with an ASV model as the target of the attack.

Spoofs can be crafted through replay, speech synthesis, and voice conversion techniques. For the replay spoofs, the adversary needs access to a sample from the voice of the *target speaker* (the person whose voice the spoof is meant to mimic and thereby use that voice to

bypass the ASV). The speech synthesis techniques require multiple audio samples of the target speaker to be able to train a model capable of synthesizing audio in the voice of the target speaker. The voice conversion techniques require multiple samples of source-target audio pairs to train a model to perform the voice conversion from the source to the target speaker.

Physical Spoofing Attacks: The physical attack (physical access attack) scenario is where a speaker's voiceprint first has to *go through a capture device* (like a microphone) *before being passed to the backend*. This scenario will involve acoustic environment variability along with capture device variability. A real-world example of a physical attack is when a user (or adversary) accesses a physical vault — that can contain items like high-value jewelry or artwork — that is secured by an ASV-based authentication. The user in this example will have to provide a voice sample at the location of the physical vault. This means that the capture device of the authentication system and the environment around the vault is chosen by the builders of the vault — whose primary goal is to ensure the security of the vault.

Logical Spoofing Attacks: The logical attack (logical access attack) scenario is in contrast to the aforementioned scenario, wherein the speaker's voice sample *can bypass the capture device of the ASV model*. An example of this is a scenario where the capture device is under the control of the user (or adversary). For example, when accessing a bank account — that is protected by an ASV-based authentication — remotely, certain banks require the user to provide a custom voice sample. This voice sample is spoken by the user and passes through the capture device — most likely a mobile phone — in the user's possession if the user is not an adversary. If the user is an adversary, they can directly pass the voice data to the communication channel, thereby bypassing the microphone.

3.2.2 Adversarial Attacks

We now elucidate the capabilities of the adversary when attacking an ASV model using adversarial attacks. To begin, the adversary requires an algorithm to generate adversarial examples to perform the attack. Then, there are two factors of variability to consider when conducting an attack.

Firstly, the type of access the adversary has to the ASV model, that is, does the adversary have black-box or white-box access? If the adversary has white-box access, they can directly backpropagate gradients through the model to generate adversarial examples. If the adversary has black-box access, they will need to generate adversarial examples from another ASV model that they have white-box access to. This is used to conduct an attack

on the *target ASV* model using the transferable property of adversarial examples between models.

Secondly, the type of access the adversary has to the *target speaker's* stored audio *template*. There are two different ways in which the adversary can target the ASV model — depending on whether the adversary has access to the speaker's audio *template* stored by the ASV model or not. The different adversary models based on template access are mentioned below:

`Read-Only-Template-Access` The adversary *has read access* to the template used by the ASV model.

`No-Template-Access` The adversary *does not have access* to the template used by the ASV model. But, they have access to a voice sample of the speaker that is distinct from the template.

There is another scenario where the adversary has read&write access to the template. We do not consider this because it falls under poisoning attacks and is out of the scope of this thesis.

3.3 Research Questions & Metrics

We will now annotate the objectives of this thesis in the following three research questions:

Research Question I : *Can a unimodal CM be augmented with visual information to improve spoofing detection rate?* To answer this question, we construct a multimodal CM by augmenting a unimodal CM with a video model that processes visual information in parallel. This approach of augmenting an existing unimodal CM is interesting because for the past few years, there have been numerous work on unimodal CM due to challenges like ASVspoof 2019 [60]. If we are able to successfully augment these types of models, then it can serve as a very simple mechanism to create multimodal models. This multimodal CM is compared with a unimodal CM to showcase its performance. The metrics used for evaluation are mentioned below:

M1.1 Equal Error Rate (EER) on test set
M1.2 Run-time of the models

Research Question II : *How do adversarial attacks affect multimodal ASV models?* To answer this, we find the best way to attack a multimodal ASV model using adversarial attacks. The metrics used for evaluation are mentioned below:

M2.1 Success rate of the adversarial attacks

Research Question III : *Can a malicious sample detector be created that can identify both spoofing and adversarial attacks against ASV models?* To answer this question, we create a novel malicious sample detector to defend against spoofing attacks from Research Question I and adversarial attacks from Research Question II. The metrics used for evaluation are mentioned below:

M3.1 Adversarial attack success rate
M3.2 True Positive Rate (TPR) on the anti-spoofing task

Chapter 4

Multimodal Spoofing Countermeasure

In Chapters 1 and 2, we have discussed about spoofing attacks and how they are defended against using a spoofing CounterMeasure (CM). But existing CMs work by using only the audio information to determine whether the given input is spoofed. Here, we research the addition of an unspoofed visual modality and identify whether any performance gains can be obtained when combining the audio and visual information to solve the task.

Here we address Research Question I from Chapter 3 which is as follows: *Can a unimodal CM be augmented with visual information to improve spoofing detection rate?* To answer this question, we construct a multimodal CM by augmenting a unimodal CM with a video model that processes visual information in parallel. This approach of augmenting an existing unimodal CM is interesting because for the past few years, there have been numerous work on unimodal CM due to challenges like ASVspoof 2019 [60]. If we are able to successfully augment these types of models, then it can serve as a very simple mechanism to create multimodal models. This multimodal CM is compared with a unimodal CM to showcase its performance. Recall that the metrics used for evaluation are:

M1.1 Equal Error Rate (EER) on test set
M1.2 Run-time of the models

To tackle this task, we replicate a unimodal CM and compare it to a novel multimodal CM. We start by explaining the dataset (Section 4.1) used to conduct the experiments followed by the preprocessing (Section 4.2) used on the dataset. Following this, we explain the methodology (Section 4.3) used to conduct the experiments. This is succeeded by the

architecture (Section 4.4) of our *multimodal* CM and the experimental results (Section 4.5).

4.1 Dataset

We use the AVspoof [15] dataset for both training and evaluating the CM models — we will hereafter refer to the dataset as AVSPOOF. AVSPOOF is an audio-visual dataset consisting of recorded videos spanning 44 speakers with a variety of different capturing devices and environmental conditions. Spoofing attacks of replay, speech synthesis, and voice conversion are carried out on the recorded audio information while the visual information is left untouched.

More specifically, we use the BTAS [26] challenge dataset which is based upon AVSPOOF — we will hereafter refer to the dataset as BTAS. They focused on the physical attack scenario while also introducing new attacks that were not a part of the training set to evaluate the effect of unknown attacks on the CM models. The new attacks focus on the scenario where the verification system is on a new device and the scenario where a voice recording is stolen or secretly captured. The training, development, and testing splits have 38580/4973, 38580/4995, and 44920/5576 spoofs/genuine data, respectively.

In Table 4.1, the various attacks from BTAS have been described. There are 4 different capturing devices (microphones) that the dataset uses, namely, the laptop microphone (AT2020USB+), phone1 (Samsung Galaxy S4), phone2 (iPhone 3GS), and phone3 (iPhone 6S). One key point to note is that following the process mentioned in the "Description" column of Table 4.1, the audio is captured by the microphone of the Automatic Speaker Verification (ASV) system. Except two attacks — replay_laptopphone2_phone3 and replay_phone2_phone3 — all of the other attacks have the ASV system placed on the *laptop*. In those two attacks, the ASV system is placed on *phone3*.

Table 4.1: Description of the attacks in BTAS. Following the process mentioned in the "Description" column of the Table, the audio is captured by the microphone of the ASV system.

Attacks	**Description**
genuine	Data that is not spoofed and emitted directly from an individual
replay_laptop	Data that is captured by the laptop's microphone and replayed by the laptop's speaker
replay_laptop_HQ_speaker	Data that is captured by the laptop's microphone and replayed by a high-quality speaker
replay_laptopphone2_phone3	Data that is captured by the laptop's microphone and replayed using the speakers of phone2
replay_phone1	Data that is captured by the microphone of phone1 and replayed using the speakers of phone1
replay_phone2	Data that is captured by the microphone of phone2 and replayed using the speakers of phone2
replay_phone2_phone3	Data that is captured by the microphone of phone2 and replayed using the speakers of phone2
speech_synthesis_physical_access	Data from the speech synthesis mechanism is played through the speakers of the laptop
speech_synthesis_physical_access_HQ_speaker	Data from the speech synthesis mechanism is played through a high-quality speaker
voice_conversion_physical_access	Data from the voice conversion mechanism is played through the speakers of the laptop
voice_conversion_physical_access_HQ_speaker	Data from the voice conversion mechanism is played through a high-quality speaker

4.2 Video Macro Alignment

With AVSPOOF being an audio-visual dataset, the audio has been curated to the level of individual utterances. But, the visuals corresponding to those utterances have not been provided. Instead, we are provided the video recordings of the complete session, that is, multiple utterances are present in a single video. So, we implement a voice activity detection-based algorithm to generate the visual input for individual utterances.

Algorithm 4: Video Alignment Algorithm for Individual Utterances

Input: Complete video *vid*,
Utterances *utts* belonging to video *vid*,
Voice Activity Detection model VAD,
Number of possible alignments *range*

Output: Visual Alignment Mapping *map*

Load pre-trained parameters of the VAD model
$voice\_activity = VAD(vid)$
foreach *utt* in the list *utts* **do**
    create empty list *possibilities*
    foreach *start* in the set *voice_activity* **do**
        /* *start* contains the beginning of a continuous utterance as detected by the VAD */
        foreach $shift$ in -range,...,range **do**
            /* $shift$ is used to pin-point specific start of *utt* in *vid* */
            add $sum(|utt - video[start - shift : start - shift + len(utt)]|)$ to *possibilities*
    map[*utt*] = argmin(*possibilities*)
return *map*

Algorithm 4 details how we extract the visuals corresponding to the audio of the utterances. A voice activity detector[1] is used to detect the *start* of a sequence of uninterrupted (not blank) audio. Then we compare the waveforms of each of the uninterrupted audios detected by the voice activity detector with each of the utterances that belong to the same video. In addition, we allow for a shift to take place on the voice activity detector's *start* that accounts for the detector generating too strict/lenient of a *start*. To ensure that the

[1] https://github.com/snakers4/silero-vad

aligment is correct, we use an automatic speech recognition module to compare the transcription of the individual audio utterance to the transcription of the utterance from the video (with the newly identified alignment).

4.3 Methodology

In the multimodal scenario, there are two inputs, namely, the audio and visual inputs. Therefore, to adapt the spoofing CounterMeasure-Automatic Speaker Verification (CM-ASV) pipeline in Figure 3.1 to the multimodal scenario, there needs to be two capture devices that are used to obtain the inputs.

As mentioned in Section 3.2.1, for physical spoofing attacks, it is assumed that the adversary can only control the input to the capture device. But, due to the lack of a realistic dataset that contains paired spoofs of visuals along with audio — recent anti-spoofing datasets only contain the audio and do not have the corresponding visuals — we use BTAS. Recall that BTAS models such an adversary for the audio input, that is, the dataset includes various types of audio spoofs. But the visuals corresponding to the audio are not spoofed. So when we conduct experiments with it, it will be under the case where the adversary can conduct a perfect replay of the visual information.

Furthermore, when adding the visual information as input to test the feasibility of a multimodal CM, the following question arises: *what visual information should be paired with the audio information?* The visual information can be broadly thought of as falling under a spectrum with two extremities. On one end of this spectrum, we have an adversary that can spoof the video to match the audio spoof — this represents the more difficult scenario to defend against ($\mathbf{ADV}_{difficult}$). On the other end of the spectrum, we have an adversary that produces visuals irrelevant to the utterance — this represents the easier scenario to defend against ($\mathbf{ADV}_{easy}$) since the visuals are not matching the audio. A realistic dataset would contain visuals falling somewhere in this spectrum. But due to a lack of this realistic dataset (as mentioned in the previous paragraph), we test on both the difficult and easier end of the spectrum so that we can establish a lower-bound and upper-bound respectively with regards to the performance of a multimodal CM. A functional multimodal model will work well against the $\mathbf{ADV}_{easy}$ setting and should not drop performance on the $\mathbf{ADV}_{difficult}$.

4.4 Model Architecture

Tasks such as speech enhancement [41; 58] and speech separation [14] have deep learning architectures that take advantage of multimodal input data to enhance performance in their tasks compared to the unimodal models. Inspired by these works, we create a multimodal model for the task of anti-spoofing that uses unspoofed visual information to jointly solve the task.

To fuse the audio and visual information, the typically used mechanism is to directly concatenate the information on a per-timestep basis. But, this is not completely correct since the information alignment between audio and video is not direct. The work by Schwartz and Savariaux [47] shows that there exists a small but clear asynchrony between the auditory events and visual events. In addition, prior to this work, the work by Chandrasekaran et al. [7] also showcased the asynchronies. We propose a multimodal model that uses attention to align the audio and visual information and solve the anti-spoofing task. Figure 4.1 gives an overview of the architecture of the multimodal CM. Each of the components — audio model, video model, fusion model, and classification model — are explained in the following sections. The input for the model is one second in length.

4.4.1 Audio Model

We use a similar audio model as Wang et al. [58]. We reduce the hidden dimension size to 256 to save on computation time. We use the Long Short-Term Memory (LSTM) so that we can conduct a local level attention in the fusion model — the reason for this will be explained in Section 4.4.3.

The input to the audio modality consists of Log Power Spectrum (LPS) computed with an input audio waveform re-sampled to 16000 samples/sec, window length of 400, hop length of 160, and FFT size of 512. We consider $FeatureExtractor()$ to perform the LPS extraction and normalization. After extraction, we have T timesteps of LPS. The audio model consists of 2 layers of LSTM that work as follows:

$$a_{1,2,....,T} = FeatureExtractor(waveform) \tag{4.1}$$

$$h_{1,2,....,T} = LSTM_1(a_{1,2,....,T}) \tag{4.2}$$

$$feats_{a1,a2,....,aT} = LSTM_2(h_{1,2,....,T}) \tag{4.3}$$

The input waveform is first converted to LPS. The output of the first LSTM layer is denoted by h_t where t is the current timestep and the input LPS is denoted by a_t. This is again fed into another later of LSTM that outputs audio features $feats_{at}$.

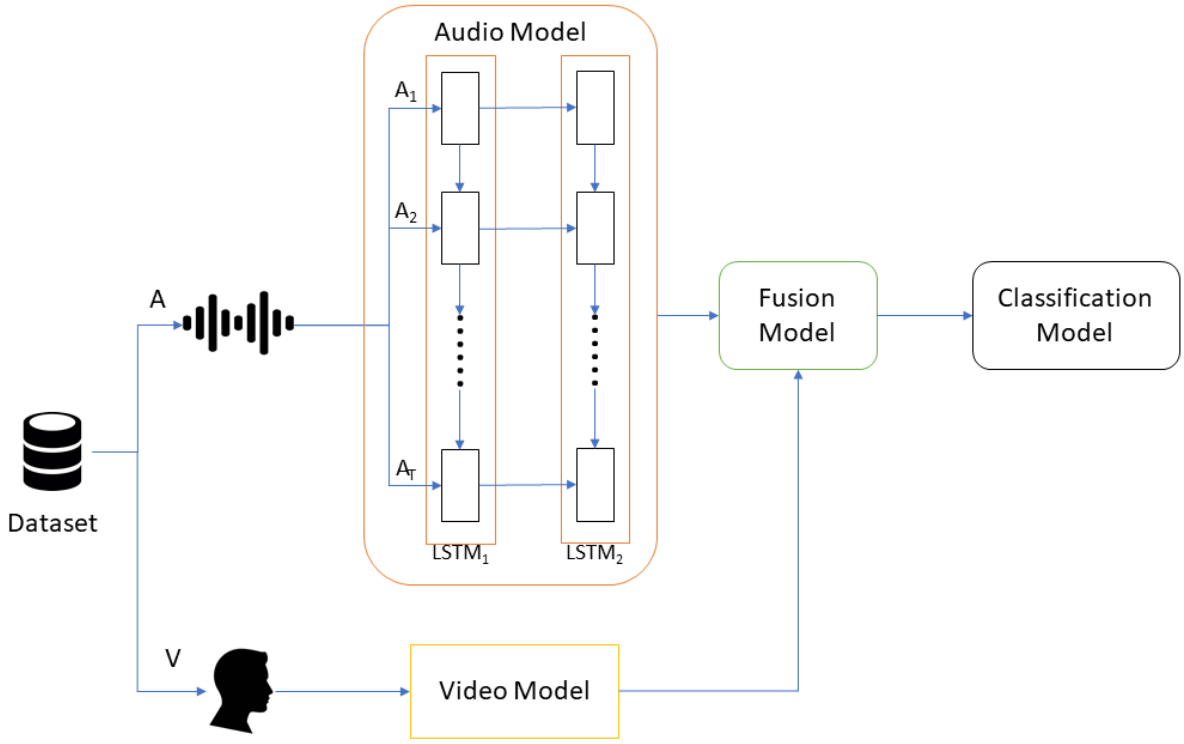


Figure 4.1: Architecture of the proposed multimodal CM. There are four components in the multimodal CM, namely the Audio, Video, Fusion, and Classification models

We also create another multimodal CM based on the unimodal *LCNN-trim-pad* from Wang and Yamagishi [59] which obtained state-of-the-art performance in the ASVspoof 2019 challenge [28].

4.4.2 Video Model

We use a similar video model as proposed in Ramesh et al. [41].

The input to the video model consists of crops of the mouth images of the speaker of size 160x160 at every timestep. The model uses a depthwise separable convolution [21] based feature extractor. The architecture for it is described on Table 4.2.

Table 4.2: Architecture of the Video Model

Component	Filters	Kernel size	Stride
SeparableConv	64	3x3	2x2
SeparableConv	64	3x3	2x2
SeparableConv	128	3x3	2x2
SeparableConv	128	3x3	2x2
SeparableConv	256	3x3	2x2
SeparableConv	256	3x3	2x2
FC-256	-	-	-

4.4.3 Fusion Model

The fusion model is responsible for combining the information from the audio and video models — the input from them are sequences of embeddings. Due to the presence of a small asynchrony between the auditory event and visual event, direct concatenation methods like Wang et al. [58] will not be the most optimal way to solve the task. We propose to use a local attention [3] mechanism to accomplish this fusion task. The attention mechanism can selectively choose the appropriate visual timesteps to attend to in a dynamic input-specific manner. Attention maps the local information into a context vector which serves as visual information that is concatenated with the audio information to pass to the audio-visual LSTM. It can be formulated as follows:

$$\alpha_{t,s} = \sigma(f(v_s, av_{t-1})) \tag{4.4}$$

$$c_t = \sum_{i=centre-k}^{i=centre+k} \alpha_{t,i} v_i \tag{4.5}$$

$$av_t = LSTM([a_t \oplus c_t]) \tag{4.6}$$

The above set of equations refers to the audio-visual LSTM used to fuse the two modalities. $\alpha_{t,s}$ is the scores used to weigh the visual output v_s and compute the context vector c_t at timesteps (t, s) of the audio and visual respectively. σ refers to the softmax function used across the local attention window. f is the scoring function used to compute relative importance scores (we use the model from Bahdanau et al. [3]), a_t is the output from the audio model, and av_t is the multimodal output. The multimodal LSTM takes as input the concatenation of audio a_t and context vector c_t. Here we use the local attention with a window size of k due to the asynchrony being local in nature. The *centre* refers to the visual frame corresponding to an audio frame if the audio-visual input were not asynchronous.

We set it to be equal to $t/4$ given that we sample the visual frames at 25 frames/sec. We also set the window size to 2.

4.4.4 Classification Model

At the end of the audio-visual LSTM, we have a sequence of features of size T. We use a Feed-Forward Neural Network (FFN) on the flattened features from the audio-visual LSTM to compute the final classification score. We use cross-entropy loss combined with adam optimizer to train the model.

4.5 Results & Discussion

Table 4.3: Results of the spoofing countermeasure models on the test set of BTAS. The models above the first horizontal dividing line are unimodal while the rest are multimodal. Each model is trained three times. The EER results here are the mean±standard-deviation of the EERs of three different models.

Model	EER	Modality	Training Adv	Testing Adv
LCNN	2.16±0.02	A	-	-
LSTM	0.93±0	A	-	-
LCNN–CNN–Fusion	2.15±0.03	AV	$\text{ADV}_{difficult}$	$\text{ADV}_{difficult}$
LSTM–CNN–Fusion(A)	1.14±0.18	AV	$\text{ADV}_{difficult}$	$\text{ADV}_{difficult}$
LCNN–CNN–Fusion	2.15±0.03	AV	$\text{ADV}_{difficult}$	ADV_{easy}
LSTM–CNN–Fusion(A)	1.14±0.18	AV	$\text{ADV}_{difficult}$	ADV_{easy}

Table 4.3 shows the EER metric of the anti-spoofing models. Each of the models are trained on the $\text{ADV}_{difficult}$ set. Each of the results in Table 4.3 is obtained from training three separate models and presenting the mean and standard deviation of the EER. This is done to showcase the result of randomness in the training process. The LCNN model is the unimodal *LCNN-trim-pad* model. For multimodal models, the naming scheme follows "Audio Model–Video Model–Fusion Model". LSTM refers to the audio model described in Section 4.4.1. CNN refers to the video model described in Section 4.4.2. Fusion(A) refers to the fusion mechanism from Section 4.4.3. So, LCNN–CNN–Fusion refers to the LCNN being used for audio processing and CNN from Section 4.4.2 for video processing and the direct concatenation of audio and visual outputs for fusing the audio and visual information.

From the results in Table 4.3, the LCNN is worse off compared to the LSTM model even though the LCNN model is made specifically for the purpose of anti-spoofing. But, this is because we use an audio duration of only 1-second as the input to both of the models. [2] This is hindering the performance of both the models and using the complete audio input will reduce this gap significantly.

Moving onto the multimodal models, first we talk about the results obtained when testing the multimodal models on the difficult end of the spectrum from Section 4.3. The key point to note is that when the visual information is added, the multimodal models either do not offer any performance improvements compared to the unimodal models in — comparing LCNN and LCNN–CNN–Fusion — or the performance drops — comparing LSTM and LSTM–CNN–Fusion(A). We can observe a huge standard deviation in the LSTM–CNN–Fusion(A) — this is because this model is influenced by randomness in training [3] thereby causing the model to converge to different points. We are uncertain on the exact cause as to why the random initialization affects LSTM–CNN–Fusion(A) model severely compared to the other ones.

Finally, when pairing the visuals obtained from the easier end of the spectrum in Section 4.3 for testing, there is no improvement in the performance of the multimodal model. In fact, when comparing the results, we can see that they are the same — the cause for this is because the gradient of the output with respect to the visual input contributes less than 1% towards the final scores. All of the above observations combined suggest that the visual information is unable to provide any performance gains in the task with the current multimodal paradigm. We believe that this is due to the nature in which the visual information has been added. We have experimented with adding visual information directly to solve the anti-spoofing task. Therefore, due to the lack of sufficient grounded supervision for the task, we believe that the visual information is unable to provide any help. For example, other approaches to including the visual information can be like Chugh et al. [8] or using explicit synchronization scores as features [9] or using adding a separate active speaker detection module in front of the CM (this would handle the scenarios where the lip is not synched with the audio).

Table 4.4 contains the runtime of the unimodal and multimodal CMs per query that has been averaged over the entire testing split. The run-time of the LSTM–CNN–Fusion(A) model is 4ms per query which is the longest run-time out of all the models. Compared to the LCNN's 0.5ms per query this is a significant overhead but not an unreasonable one

[2] Since our results indicated that the method of adding visual information is not helpful, increasing this will not cause our multimodal models to improve.

[3] Randomness originates due to weight initialization and data sampling.

Table 4.4: Run-Time of the CMs in milliseconds (ms).

Model	Run-Time
LCNN	0.5
LSTM	2.0
LCNN-CNN-Fusion	0.6
LSTM-CNN-Fusion(A)	4.0

considering that we use LSTMs for the audio model. If we were to use the LCNN for the audio model as in LCNN–CNN–Fusion, the runtime is 0.6ms per query. This is an acceptable overhead due to the addition of a new fusion component and the processing of a new modality. When comparing LSTM–CNN–Fusion(A) with its unimodal model LSTM, the runtime is within the acceptable range as well.

To summarize, we have answered Research Question I by showing that a multimodal CM cannot improve performance over a unimodal CM in this paradigm. We approach the construction of a multimodal CM using the augmentation approach due to existing works on unimodal CMs, that is, if successful, this can provide an easy mechanism to integrate visual information into unimodal models. But, the results indicate that the direct integration of visual information does not provide suitable information for the task. If the input from both modalities were to be integrated using a different method like [8] this conclusion may not hold but we leave it to future work to explore these kinds of methods.

Chapter 5

Adversarial Attack on Multimodal Speaker Verification

In Chapters 1 and 2, we have discussed adversarial attacks and how they present an attack vector different from spoofing attacks. But existing works on the effects of adversarial attacks on Automatic Speaker Verifications (ASVs) have not considered *multimodal* ASVs.

Here we address Research Question II from Chapter 3 which is as follows: *How do adversarial attacks affect multimodal ASV models?* To answer this, we find the best way to attack a multimodal ASV model using adversarial attacks. Recall that the metric used for evaluation is:

M2.1 Success rate of the adversarial attacks

To answer this question, we replicate two *unimodal* ASV models, two *multimodal* ASV models and conduct adversarial attacks on the multimodal models using the unimodal and multimodal models. We start by explaining the dataset (Section 5.1) used to conduct the experiments followed by the ASV models (Section 5.2) used. This is succeeded by the methodology (Section 5.3) of how we conduct our experiments and their results (Section 5.4).

5.1 Dataset

We create the ASV models by training them on the VoxCeleb1 [36] dataset — we will hereafter refer to the dataset as VOXCELEB1. VOXCELEB1 is an audio-visual dataset

consisting of videos curated from YouTube following a pipeline consisting of face tracking, active speaker verification, and face verification. The curated videos are gender-balanced, and consists of speakers from a variety of ethnicities, ages, and accents but are unilingual — the language spoken is English-only. The audio tracks from the videos consist of a variety of acoustic environments along with various degrading factors like noise. VOXCELEB1 has two separate splits — the first one consists of 1211 speakers and ∼145,000 utterances while the second consists of 40 speakers and ∼4,800 utterances.

But, some videos are not available on YouTube at the time of this thesis thereby leading to a reduction in the number of utterances. The reduced version of the first set consists of 1211 speakers and ∼122,000 utterances while the reduced second set consists of 40 speakers and ∼4,100 utterances.

It is curated such that there is no overlap between the speakers from the first and second sets. The first set is used for training while the second is used for testing. We curate a custom set for validation from the existing first set by creating 12 positive and negative pairs for each speaker in the first set. These pairs are used as the validation set. The remaining data from the first set is used as the training set.

5.2 Automatic Speaker Verification models

We replicate four ASV models to assess robustness against adversarial attacks — x-vector [50][1], 1D-CNN from Shon et al. [49], online person verification [48], and multiview [45]. From here on, they will be referred to as X-Vec, 1D-CNN, OPV, and MV respectively. X-Vec and 1D-CNN are unimodal models while OPV and MV are multimodal models.

Table 5.1 contains our evaluation of the ASV models on the testing split of VOXCELEB1 dataset — **Equal Error Rate (EER)**, **F1-Score**, **Precision** and **Recall**. We can observe that the multimodal models perform significantly better than the unimodal ones as previous works have shown.

For comparison, Table 5.1 also shows the EER scores ($\mathbf{EER}_{cmp}$) taken from the corresponding papers as cited in the table. We've added them so that the readers can get a sense for reference but at the same time they should be mindful that the scores cannot be compared directly due to a variety of reasons as mentioned below.

Due to missing videos in VOXCELEB1's test set as mentioned previously, the scores are bound to be different across all of the models. 1D-CNN's comparison score is obtained from

[1] https://github.com/cvqluu/TDNN

Table 5.1: Evaluation of the 4 ASV models on the test set of VOXCELEB1. The models above the horizontal dividing line are unimodal while the rest are multimodal.

Model	EER	EER_{cmp}	F1-Score	Precision	Recall	Modality
X-Vec	12.30	11.3 [49]	0.88	0.89	0.86	A
1D-CNN	13.20	8.4 [49]	0.86	0.89	0.84	A
OPV	7.55	5.29 [48]	0.92	0.91	0.94	AV
MV	8.42	1.4 [45]	0.92	0.93	0.91	AV

a model trained with data augmentation which is known to boost performance [50]. OPV's comparison score is obtained from a model trained with utterances from the VoxCeleb2 [10] dataset — which is multiple times larger than VOXCELEB1. The score is obtained from testing on the test set of VoxCeleb2 as well. MV's comparison score is obtained from a model trained on VoxCeleb2 and tested on VOXCELEB1.

5.3 Attack Methodology

Figure 5.1 showcases the pathways for the forward and backward passes of multimodal ASV models — pathways for unimodal ASV models can be directly extrapolated by removing one of the modalities. The blue lines show the forward pathway (used to compute output) while the orange lines show the backward pathway (used to compute gradients).

We attack the ASV models with adversarial examples generated using the various algorithms mentioned in Section 2.3.1. These generation algorithms rely on computing gradients $Grads_{A_x}, Grads_{V_x}$ corresponding to the inputs A_x, V_x respectively. They use a *source* model to obtain gradients. The attacks are conducted under the white-box and black-box settings — the attack is white-box when the *target* model is the same as the source model while the attack is black-box when the target model is different from the source model. The black-box attacks are also referred to *transfer attacks* since the adversarial examples generated for a source model are *transferred* to a target model different from the source.

We conduct the attacks for adversary models `Read-Only-Template-Access` and `No-Template-Access` as defined in Section 3.2.2. Recall that the adversary can only change the inputs corresponding to one of the input pairs of speaker data (Figure 5.2) in both adversary models. According to those algorithms, we consider the inputs from the attacker x as *candidate inputs* (A_c, V_c) and aim to perturb these. For `Read-Only-Template-Access`, we denote inputs from the victim y as *template inputs* (A_t, V_t) since they originate from the

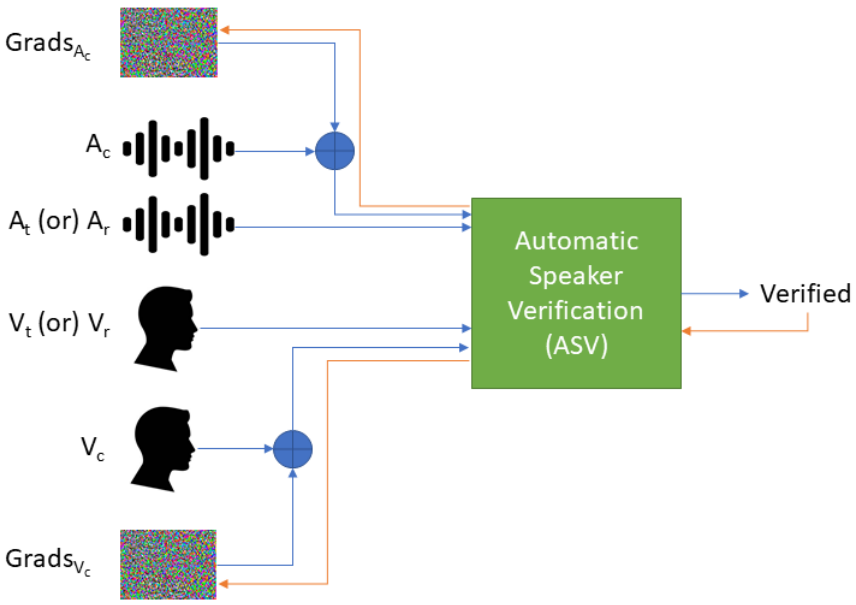


Figure 5.1: Depiction of how adversary models compute the perturbation using a multi-modal ASV model. For `Read-Only-Template-Access`, we use (A_c, V_c) and (A_t, V_t) to compute the perturbations. For `No-Template-Access`, we use (A_c, V_c) and (A_r, V_r) to compute the perturbations.

template that has been obtained by the adversary. On the other hand, for `No-Template-Access`, we denote the inputs from the victim y as *reference inputs* (A_r, V_r) since they originate from the same identity as the template but a different utterance. This has been depicted in Figure 5.1. In both adversary models, the perturbed version of the candidate input is compared with the template input to compute robustness of the ASV. This has been depicted in Figure 5.2.

The resulting robustness of the models is evaluated using the *attack success rate* metric which corresponds to the percentage of adversarial examples that are wrongly classified.

We create adversarial examples for `Read-Only-Template-Access` using ∼1000 pairs of negative — therefore, in this case, the attack success rate will be equivalent to the false positive rate — testing data from the VOXCELEB1 dataset. The pairs are chosen randomly from a subset of testing data that all of the four ASV models correctly classify. The testing pairs for `No-Template-Access` is obtained from the test set following the same criterion. We run this experiment four times — each time with a different set of input pairs from the test set. Each of the utterances from the test set act as a candidate input once across

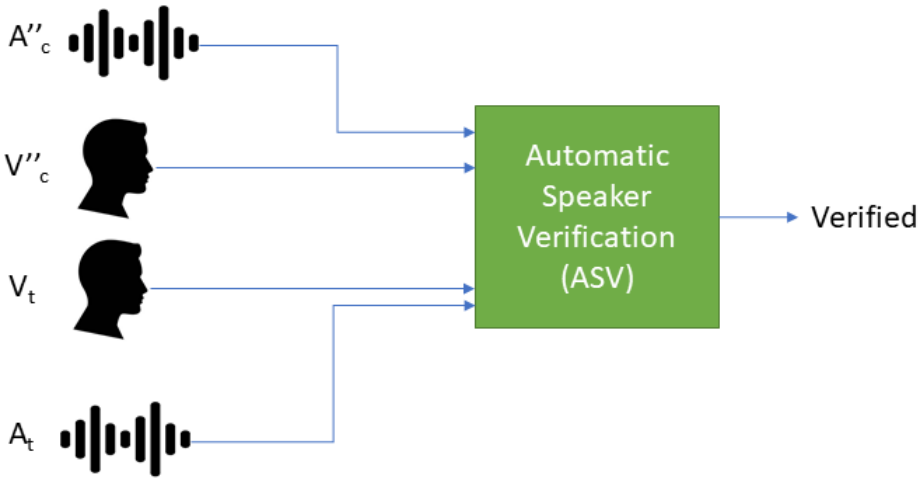


Figure 5.2: Depiction of how the robustness of ASV models is tested. A_c'' and V_c'' are the perturbed audio and visual inputs. A_t and V_t are the templates of the victim y.

the four sets with the utterances being evenly divided across the four sets.

5.4 Results & Discussion

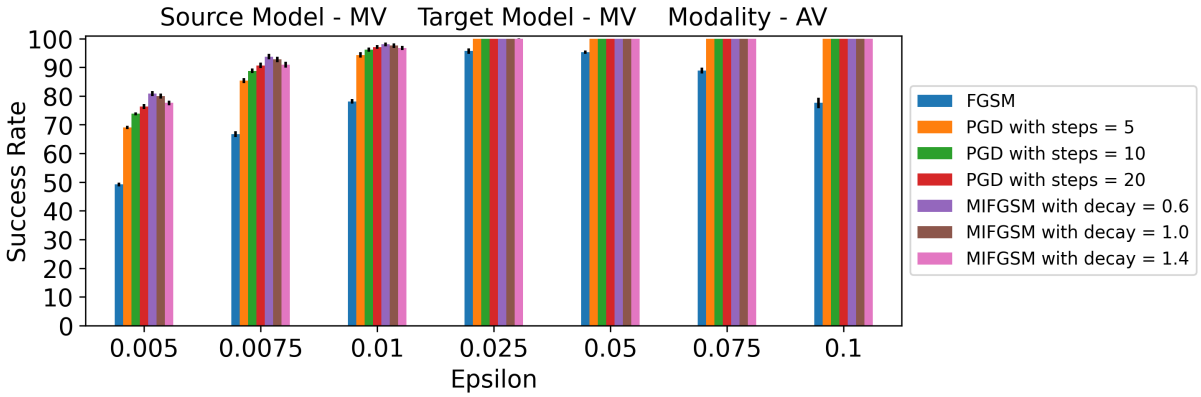


Figure 5.3: Figure showing the white-box attack targeting the audio-visual modality on MV for `Read-Only-Template-Access`

Here, we will present the distribution of attack success rates (y-axis) across different

perturbation degrees (x-axis — epsilons) in different graphs. Each graph will correspond to a specific permutation of (source model, target model, and modality attacked) which will be mentioned at the top of the graph. Also, the graphs will have bars with different colors corresponding to different attacks. *We use different y-axis ranges for each graph.* We will refer to attacks that target both the audio & visual modalities as *targeting the audio-visual modality.*

Figure 5.3 is used to show the effectiveness of different types of adversarial example generation algorithms. We were only able to test for a few variants but given additional time and resources an exhaustive search of the attacks' hyper-parameters can be conducted. We can generally observe from Figure 5.3 that the adversarial examples generated using Fast Gradient Sign Method (FGSM) have success rates that are below those of multi-step algorithms like Projected Gradient Descent (PGD) and Momentum Iterative Fast Gradient Sign Method (MIFGSM). Also, among the variations of the PGD algorithm, we can observe that the one with 20 steps is the best. Similarly, among the variations of the MIFGSM algorithm, we can observe that the one with a decay of 0.6 is the best. Although Figure 5.3 is for a specific case, we either observed the same general trend in other settings or observed that there is not a significant difference in attack success rates between the different versions of the attacks. Under certain scenarios this can change for which we will point them out directly in the discussion. For the sake of clarity, we only present the most relevant graphs to illustrate the discussion in this chapter, but all results are presented in Appendix A.

5.4.1 Results for adversary model `Read-Only-Template-Access`

Here, we will discuss the results for adversary model `Read-Only-Template-Access`.

White-box Attacks: The white-box attacks on the multimodal OPV are presented in Figure 5.4. We can directly see that attacking the audio-visual modality provides the highest success rate at a lower perturbation. Interestingly, the white-box attack on the visual modality (Figure 5.4b) has a success rate closer to that of the white-box attack on the audio-visual modality (Figure 5.4c), that is, attacking the visual modality can provide an attack which is very close to the best attack. The attack on the audio modality (Figure 5.4a) on the other hand has a much lower success rate. This indicates that the model has a significant reliance on the visual modality compared to the audio modality. This over-reliance on the visual modality is more highly pronounced for MV whose results are in the Appendix A. Another interesting point is that attacking the audio modality alone is unable to achieve complete success (100%) as opposed to attacking the visual modality. In

attacking the visual modality with FGSM, we can see that the success rate increases upto a certain epsilon and then starts decreasing with higher epsilons. This happens because the direction of the gradients change more drastically as the perturbation size grows [13]. On the other hand, attacking the audio modality with FGSM shows that a significantly lower perturbation is required when observing that the lowest perturbation gives the highest success rate. When attacking the audio modality on OPV, a decay of 1.0 is better than 0.6 and so we refer the readers to that results in the Appendix A. Also, the success rates increase with the number of steps as it should be under this scenario. Similar patterns can be observed across the white-box attacks on the MV presented in the Appendix A.

Black-box Attacks: In black-box attacks where the unimodal models serve as the source model to attack multimodal models, — Figure 5.5 shows the attacks targeting OPV — the attack success rate are generally less than 10% which is significantly lower compared to the white-box attacks. When 1D-CNN attacks the OPV, the success rates are higher (compared to when X-Vec attacks OPV) due to them sharing a similar architecture. This tells us that when attacking multimodal models, a transfer attack from a unimodal model is not sufficient to gain high success rates.

In black-box attacks where the multimodal models are the source models — Figure 5.6 shows the attacks targeting OPV — the attack success rate are lower compared to the white-box attacks. When only the audio modality is perturbed (Figure 5.6a), the attack success rates are comparable to the attacks generated from the unimodal models. But when the visual modality is perturbed (Figure 5.6b), the attack success rates are significantly higher. Similarly, when the audio-visual modality is perturbed (Figure 5.6c), the success rates (which are as high as 40%) are higher than just perturbing the audio modality (Figure 5.6a). Another point to note is that when the target of the black box attack is MV (results in Appendix A), the highest attack success rates is less than 10%. This indicates that the transferability is lower from OPV to MV. This is due to MV requiring 2 visual frames for each second of input but OPV requires only 1 visual frame for each second of the input. We conducted a black-box attack by padding zeros when transferring from OPV to MV.

5.4.2 Results for adversary model `No-Template-Access`

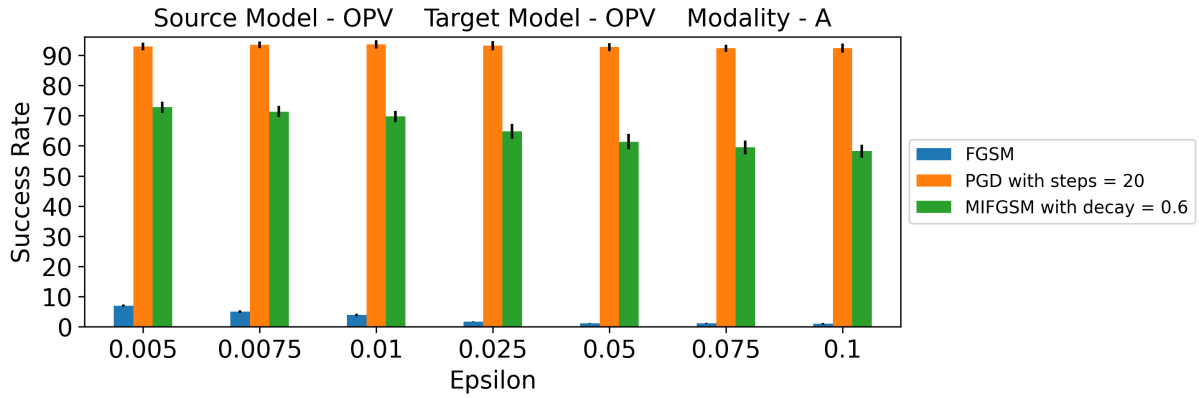
We will now discuss the results for adversary model `No-Template-Access` which is a more difficult scenario to succeed in attacking.

White-box Attacks: There is a severe reduction in white-box attack success rates under this adversary model as compared to `Read-Only-Template-Access`. Figure 5.7 shows the

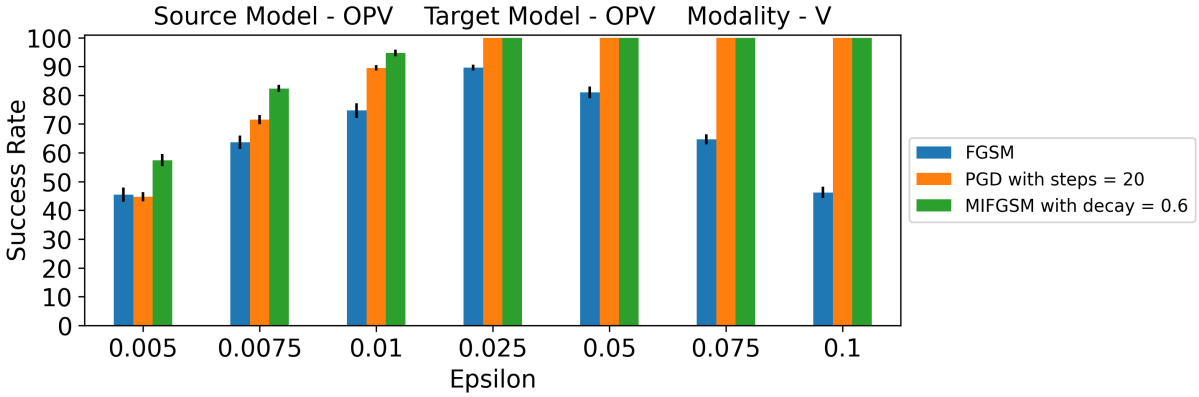
results when targeting the audio-visual modality. The maximum success rate is around 90% whereas in `Read-Only-Template-Access` it was 100%.

Black-box Attacks: Similar to the white-box scenario, there is a severe reduction in success rates. In fact, in this hard adversary model, the attack success rates are less than 10% when targeting the audio-visual modality as depicted in Figure 5.8.

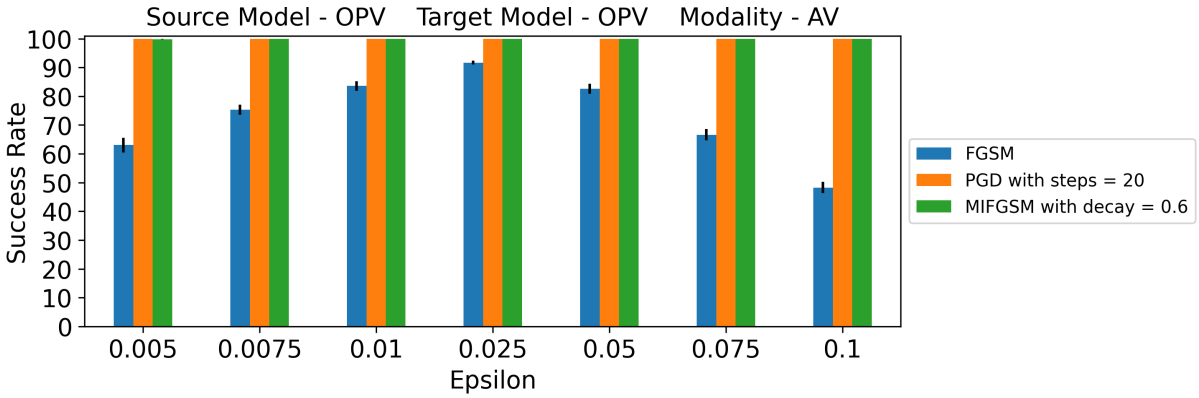
To summarize, we have answered Research Question II by showing adversarial attacks do affect multimodal ASVs by targeting the audio-visual modality — they are always the best performing attack compared to unimodal attacks across the board. But under the harder `No-Template-Access`, the attacks are not highly successful.



(a) Targeting the audio modality.

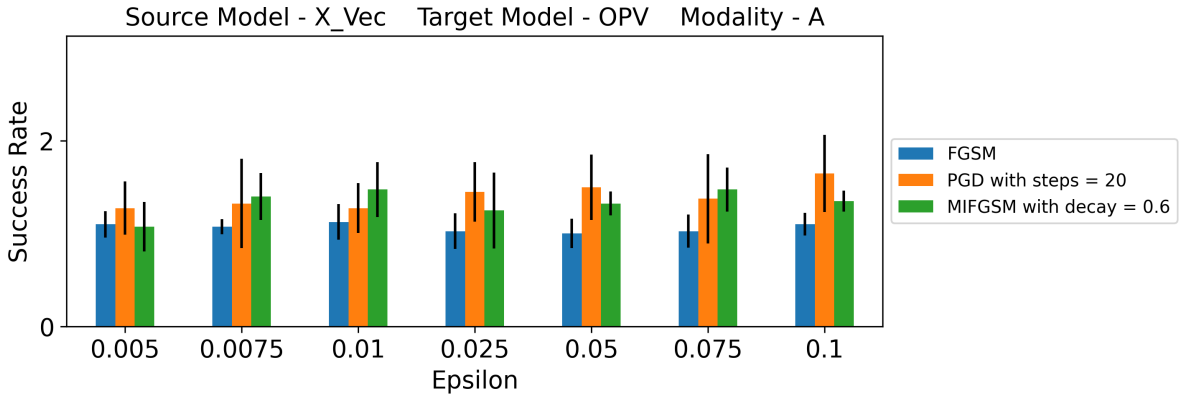


(b) Targeting the visual modality.

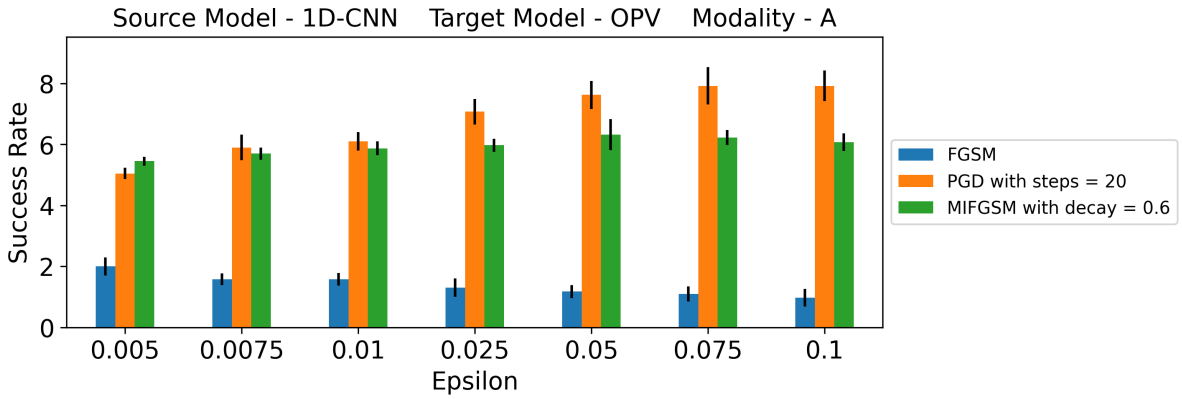


(c) Targeting the audio-visual modality.

Figure 5.4: White-box attack on OPV for `Read-Only-Template-Access`

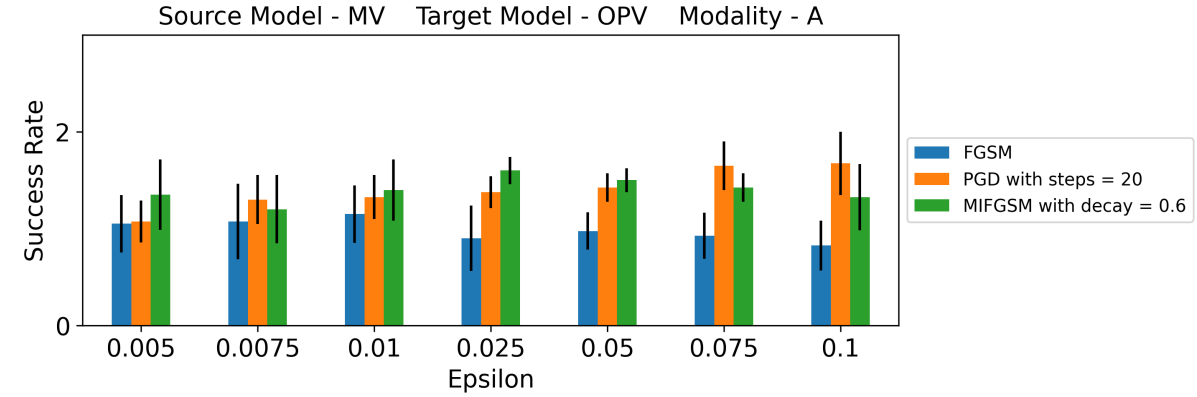


(a) Targeting the audio modality by transfer from X-Vec.

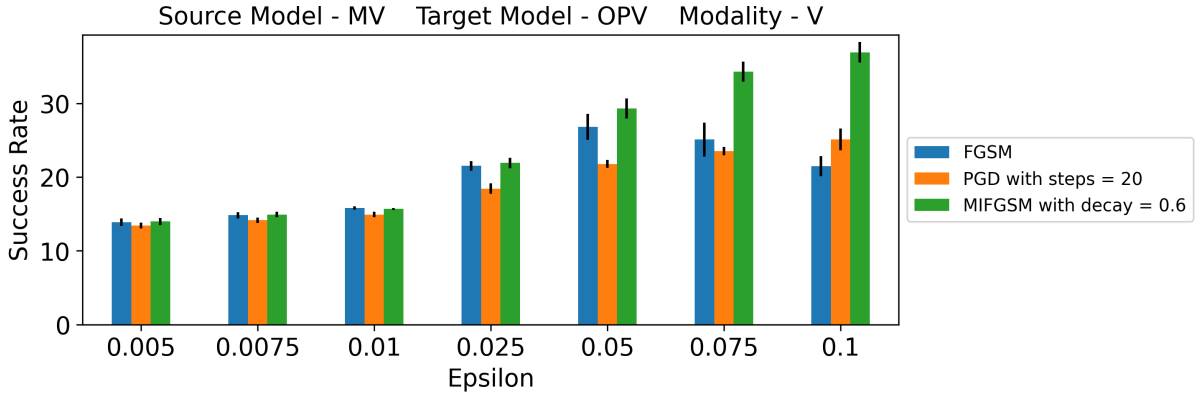


(b) Targeting the audio modality by transfer from 1D-CNN.

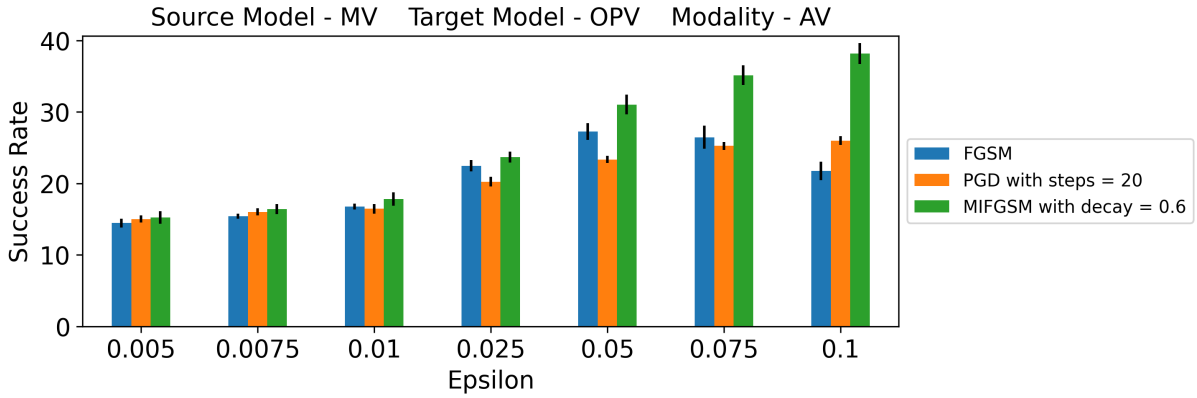
Figure 5.5: Black-box attack targeting OPV where source models are unimodal for Read-Only-Template-Access



(a) Targeting the audio modality by transfer from MV.



(b) Targeting the visual modality by transfer from MV.



(c) Targeting the audio-visual modality by transfer from MV.

Figure 5.6: Black-box attack targeting OPV where source models are multimodal for Read-Only-Template-Access

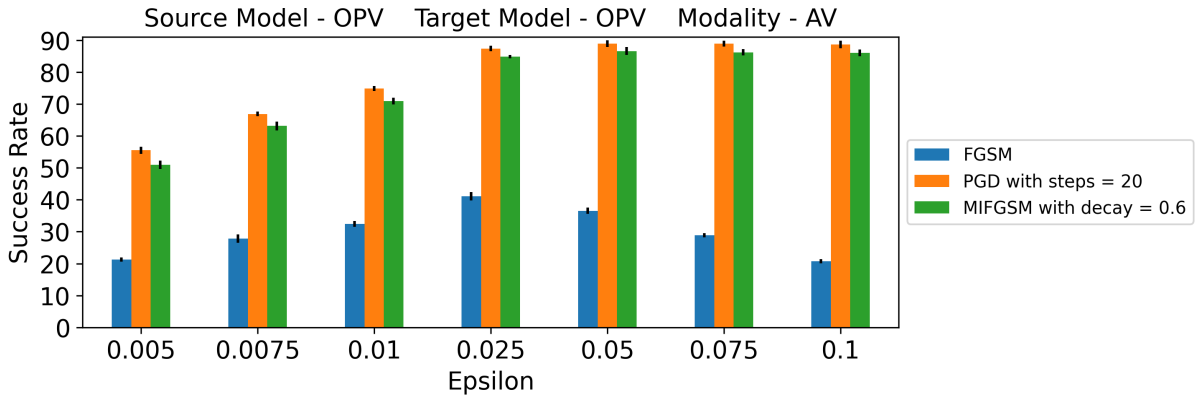


Figure 5.7: Figure showing the white-box attack targeting the audio-visual modality on OPV for `No-Template-Access`

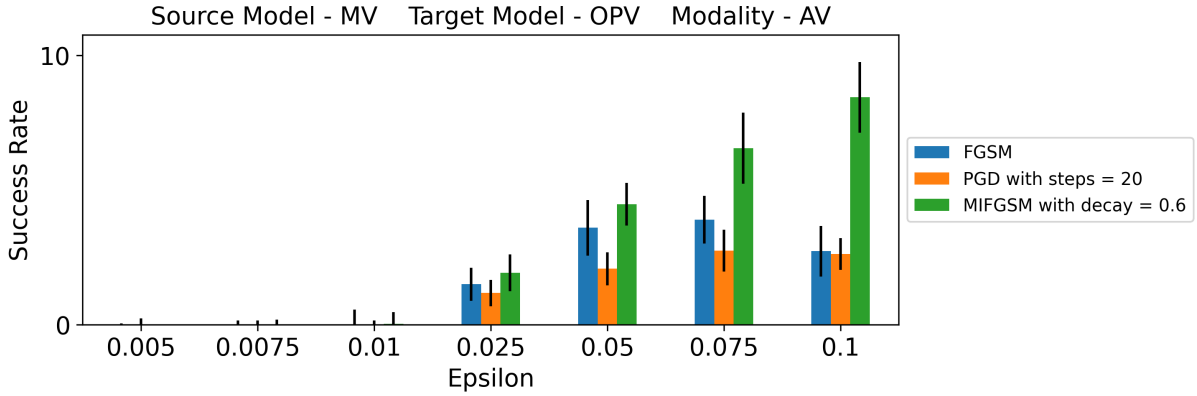


Figure 5.8: Figure showing the black-box attack targeting the audio-visual modality by transfer from MV to OPV for `No-Template-Access`

Chapter 6

Malicious Sample Detector

In Chapter 4, we showed that multimodal spoofing CounterMeasures (CMs) created by augmenting a unimodal CM does not give any performance improvements. In Chapter 5, we demonstrated that multimodal models are heavily reliant on the visual modality and by targeting this the attacker can get adversarial attack success rates that are close to attacking both the modalities (which gives the highest success rates). In this chapter, we propose to combine the functionality of adversarial example detection into the CM. This will allow the CM to become a multi-purpose defense mechanism by defending against both spoofs and adversarial examples *that target the Automatic Speaker Verification (ASV).*

Here, we address Research Question III from Chapter 3 which is as follows: *Can a malicious sample detector be created that can identify both spoofing and adversarial attacks against ASV models?* To answer this question, we create a novel malicious sample detector to defend against spoofing attacks from Research Question I and adversarial attacks from Research Question II. Recall that the metrics used for evaluation are:

M3.1 Adversarial attack success rate
M3.2 True Positive Rate (TPR) on the anti-spoofing task

To answer this question, we first explain the dataset used for the experiments (Section 6.1). Then we explain the methodology (Section 6.2) of how we structure our experiments followed by the results (Section 6.3) of the experimentation.

6.1 Dataset

There are two datasets used in this Chapter. First is the BTAS used to train the CM models. This dataset has been introduced in Section 4.1. The other dataset used is the VOXCELEB1 used to train the ASV models. This dataset has been introduced in Section 5.1.

We generate adversarial examples using the same methodology as in Chapter 5. The generated examples originate from ∼1000 pairs of negative testing data from VOXCELEB1 (same as in Chapter 5). The pairs are chosen randomly from a subset of testing data that all of the four ASV models correctly classify.

6.2 Methodology

One of the mechanisms used to make a model robust to adversarial attacks is adversarial training. But, the task that we are trying to achieve is to have the CM detect and stop the adversarial attacks *targeting the ASV models*. If we were to implement adversarial training on the CM models, they will gain robustness to adversarial attacks *that target the CMs* — which is not what our goal is. Whether this model will be resistant to adversarial examples targeting ASVs is not determined. So, we do not use adversarial training to achieve our goal.

Instead, we propose to use feature squeezing (explained in Section 2.3.2) to detect the adversarial examples. This technique does not intrinsically make the model robust, but rather detects the adversarial examples — it uses the difference in predictions between the "squeezed" and original inputs to detect the adversarial example. When an input is detected as adversarial, it is blocked without being passed to the CM — this counts as an instance of a successful detection. If an adversarial example manages to bypass this defense, but the CM classifies the input as a spoof, we consider this also as an instance of successful detection since the end result is a blocking of the input without any changes to the pipeline. As for measuring TPR on the anti-spoofing task, it is measured from the probability vector that passes the feature squeezing defense, that is, if the defense blocks a legitimate example, it counts as a false classification.

Figure 6.1 depicts how the feature squeezing mechanism is added onto the CMs. The threshold required for the feature squeezing mechanism to detect whether a given input is adversarial or not is obtained as explained in the remainder of the paragraph. The threshold is set as the Equal Error Rate (EER) between adversarial examples and legitimate

examples. In order to generate adversarial examples for the identification of the threshold, we assume that the defender has access to some ASV models — we call these *seen models* since the defender and thereby the feature squeezing mechanism will know the adversarial examples generated by these models. We set X-Vec and MV as the seen models. The adversarial example generation algorithm for the determination of the threshold is the PGD algorithm with steps of 10 and an epsilon of 0.025. We use 1000 pairs of negative data (we aim to make false acceptances by using adversarial perturbations) from VOXCELEB1's dev set for the aforementioned purpose. The legitimate examples come from the dev set of BTAS and VOXCELEB1.

During testing, the adversarial examples are generated from ASVs using the MIFGSM (decay of 0.6) algorithm with different epsilons. They are generated from VOXCELEB1's test set.

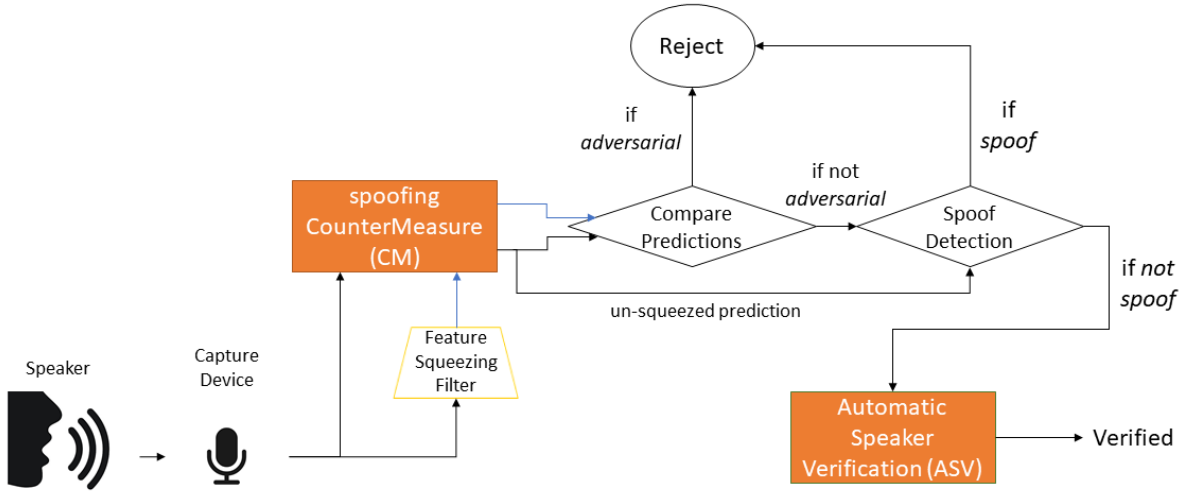


Figure 6.1: Feature squeezing pipeline when combined with spoofing countermeasures.

6.3 Results & Discussion

Tables 6.1 and 6.2 show the adversarial attack success rate of the spoofing CounterMeasure-Automatic Speaker Verifications (CM-ASVs) pipeline with and without any defense. The defense consists of a CMs which is augmented with *median* feature squeezing defense (explained in Section 2.3.2). The models used for generating the adversarial examples are the

ASVs OPV and MV introduced in Chapter 5 — hereafter they will be collectively referred to as *source ASVs*. The CMs used to detect the adversarial examples are LCNN, LSTM, LCNN–CNN–Fusion and LSTM–CNN–Fusion(A) — hereafter they will be collectively referred to as *target CMs*. LCNN and LSTM are unimodal models while LCNN–CNN–Fusion and LSTM–CNN–Fusion(A) are multimodal models.

Each of the cells in Tables 6.1 and 6.2 correspond to an adversarial attack success rate obtained from a source ASV which generates the adversarial examples and a target CM (with or without defense). In addition, for computing the attack success rates, the target CM will be combined with an ASV (*target ASV*), and so, the adversarial attack success rate being depicted here is "out of the adversarial examples that have already bypassed the ASV what percentage of them are able to bypass the CM?". The adversarial scenario is *multimodal black-box* attacks under `Read-Only-Template-Access`. So, for example, Table 6.1 corresponds to the success rates of attacking a CM-ASV pipeline where the target CM is depicted in the column 'Target Model', the target ASV is the OPV (since we stated that the scenario will be multimodal black-box attacks) and the source ASV generating the adversarial attacks is MV. The results from each cell come from running the experiment four times — each time with a different set (a set consists of 1000 examples the same as Chapter 5) of adversarial examples — and presenting the mean and standard deviation of the adversarial attack success rates. Keep in mind that out of the examples in each set, the number of examples that are successful against an ASV will be less than or equal to the set size and *the percentages being depicted in the Tables are from these examples that are successful against an ASV* — so a value of 100% refers to the examples completely bypassing the target CM and ASV.

Without any defense, in smaller perturbations, none of the target CMs have any significant capability to block adversarial attacks. This indicates that the CMs themselves are unable to block adversarial attacks targeting the ASVs. But, under larger perturbations, the CMs show some capability to block — for example, the success rate is lowered to 75.3 in Table 6.1, 51.0 in Table 6.1. This is surprising because they have not been exposed to the adversarial examples at this point at all (more on perturbation size in the penultimate paragraph of the Chapter). Recall that attack on the target CMs pose a cross-task and cross-dataset black-box attack scenario. So, under these conditions, the transferability seems to be low with increasing epsilon for the unimodal models (a similar observation was made in Li et al. [31] for black-box attacks that with increasing perturbations, the success rate decreases). On the other hand, when a defense is integrated into any of the CMs, the success rate is significantly lowered. This indicates that the CM models can provide effective information for feature squeezing even though the target of the attack is the ASV. Even in this scenario, the larger the perturbation, the better the ability of the CM to detect

Table 6.1: Adversarial attack success rate (%) on the CM-ASV pipeline. The success rate is presented as the mean±standard-deviation across the success rates of the four runs. Each attack has a source ASV from which it is generated and a **Modality** it is perturbing. Each defense mechanism consists of a target CM and the attack success rates presented here are the percentage of examples that bypass the target CM and target ASV out of the examples that bypass the target ASV. For this scenario the target ASV is the OPV. We use different colors to indicate the best (lowest) and worst (highest) metrics (based on mean) for each epsilon value. We indicate all models without defense using underline.

Source Model		MV		
Perturbed Modality		AV		
Epsilon		0.1	0.025	0.005
Target Model	**Input Defense**			
LSTM	no	75.33±1.96	94.43±2.46	98.15±0.94
	yes	2.02±0.49	25.11±3.27	54.24±4.15
LCNN	no	51.0±1.79	76.07±2.01	91.75±1.21
	yes	2.97±0.5	20.28±2.33	67.85±1.65
LSTM–CNN–Fusion(A)	no	78.77±1.04	96.09±1.95	97.67±0.8
	yes	0.39±0.21	16.04±2.04	42.93±5.12
LCNN–CNN–Fusion	no	99.27±0.48	97.58±1.27	97.04±0.71
	yes	18.45±1.12	19.98±2.04	66.57±4.14

the adversarial examples. This is interesting because the larger the perturbation, the more effective the black-box attack is in Chapter 5. The likely cause of this is because at higher perturbations when targeting the ASV, the squeezed representation's output used by the CM diverges vastly as compared to lower perturbations and since we chose a squeezing threshold using a perturbation of 0.025, higher perturbations than this are easier to detect while lower perturbations are comparatively difficult.

But, this adversarial attack detection comes at the cost of an reduction in TPR as depicted in Table 6.3. So the attack success rates must be compared in tandem with the corresponding results from Table 6.3. So even if a model gives a lower success rate, if the TPR is also reduced proportionally, then we cannot conclude that such a model is better — as is the case with LSTM–CNN–Fusion(A). The results from Table 6.3 correspond to the TPR of the target models when the input is from the legitimate samples from BTAS and VOXCELEB1 test set. We can see from Table 6.3 that there is as much as a 40% reduction in the TPR of legitimate samples which is significant. This is due to the threshold selection criteria we have outlined in Section 6.2. We have chosen the threshold based on the EER

Table 6.2: Adversarial attack success rate (%) on the CM-ASV pipeline. The success rate is presented as the mean±standard-deviation across the success rates of the four runs. Each attack has a source ASV from which it is generated and a **Modality** it is perturbing. Each defense mechanism consists of a target CM and the attack success rates presented here are the percentage of examples that bypass the target CM and target ASV out of the examples that bypass the target ASV. For this scenario the target ASV is the MV. We use different colors to indicate the best (lowest) and worst (highest) metrics (based on mean) for each epsilon value. We indicate all models without defense using underline.

Source Model		OPV		
Perturbed Modality		AV		
Epsilon		0.1	0.025	0.005
Target Model	**Input Defense**			
LSTM	no	68.84±4.27	95.18±3.27	100.0±0.0
	yes	3.57±1.89	30.97±6.05	60.87±2.71
LCNN	no	68.39±5.36	88.38±1.26	96.22±1.26
	yes	4.17±1.48	35.95±4.48	71.74±3.78
LSTM–CNN–Fusion(A)	no	71.62±4.81	95.31±1.13	100.0±0.0
	yes	0.76±0.76	20.58±4.17	47.01±4.74
LCNN–CNN–Fusion	no	100.0±0.0	99.63±0.65	98.7±1.43
	yes	50.93±4.17	35.56±6.76	72.42±3.11

between two classes of data — since we don't prioritize (keeping low) False Acceptance Rate (FAR) or False Rejection Rate (FRR). If we had chosen a threshold based on a prioritizing FRR, then the success rates in Tables 6.1 and 6.2 will be higher and vice-versa if we prioritize FAR.

Now, we proceed to talk about the effectiveness of such a detector. Recall that the CMs in this chapter are not significantly reliant on the visual information to detect the spoof. Also, a high perturbation such as 0.1 is not required to succeed in attacking audio modality of the ASV as can be observed in the white-box attacks in Figure 5.4a [23]. And such a high perturbation makes it easy for the audio CM to detect them. So, this defense can be significantly evaded if the attacker only perturbed the visual modality since the CM used is not reliant on the visual information. On the other hand, this defense is effective if the attacker uses a very high audio perturbation.

To summarize, we have answered Research Question III by showing that while CMs can block adversarial and spoofing attacks targeting the ASVs, this detection rate come at a significant cost in the form of a reduction in TPRs.

Table 6.3: Table showing the TPR (%) of the CMs from Tables 6.1 and 6.2 on the test set of BTAS and VOXCELEB1. We use different colors to indicate the best (highest) and worst (lowest) metrics (based on mean) for each epsilon value. We indicate all models without defense using underline.

Target Model	**Defense**	BTAS	VOXCELEB1
LSTM	no	99.5	98.5
	yes	61.2	75.1
LCNN	no	98.1	90
	yes	84.4	63.2
LSTM–CNN–Fusion(A)	no	99.4	98.5
	yes	65	67.8
LCNN–CNN–Fusion	no	99.3	97.1
	yes	78	61

Chapter 7

Related Work

In this chapter, we describe prior research related to some of the topics in this thesis or topics that are closely linked with the work we have done in this thesis.

Speaker recognition: The Gaussian Mixture Model (GMM) when combined with a Universal Background Model (UBM) is called "GMM-UBM". It is one of the seminal works that propelled the speaker recognition field forward [43] with it being highly effective. Beyond that, the i-vector model [12] reduces the embeddings received from the GMM-UBM model through factor analysis and models the channel and speaker variability in a combined manner. Following this came the introduction of the neural networks into the GMM-UBM models leading to Deep Neural Network (DNN) i-vector models [73]. Following this came the introduction of DNN embedding-based models like x-vector [50] where a form of the hidden layer of the DNN was used as the embedding vector for verification models. Finally, end-to-end models like online person verification systems [48] which do not require a separate backend score computation mechanism were introduced. They use an attention mechanism to fuse multimodal information. Sarı et al. [45] proposes a novel approach to handle the scenario of missing information from modalities in speaker verification by cross-modal testing. Shon et al. [49] proposes a 1D Convolutional Neural Network (CNN) to obtain speaker embeddings and modifies it to produce frame-level speaker embeddings for analysis. We implement Snyder et al. [50], Shon et al. [49], Shon et al. [48], and Sarı et al. [45] in Chapter 5 to conduct adversarial attacks against these models.

Other multimodal biometric tasks: The work by Wen et al. [62] proposes a new approach to solve the cross-modal matching of faces and voices by using multi-task supervision to obtain embeddings. They supervise the training of the model through the use of multiple goals such as gender, nationality, and identity. Tao et al. [52] augmented the performance

on the speaker recognition task using a cross-modal network that matches voices to faces. They use cosine similarity to train the network to recognize associated voices and faces. Both of these networks use the embeddings obtained after training to perform the final task. Our work is based on the task of Automatic Speaker Verification (ASV).

Adversarial attacks on speaker verification: Kreuk et al. [27] conducted adversarial attacks to evaluate their effectiveness in an end-to-end ASV system. They also conducted cross-dataset and cross-feature attacks using a waveform reconstruction algorithm. Li et al. [31] evaluated adversarial attacks on the GMM i-vector and x-vector models. They also conducted cross-model attacks and showcased their transferability from i-vector to x-vector models. Li et al. [30] developed an independent network to detect adversarial examples as a defense mechanism. Wu et al. [66] investigated a self-supervised pre-trained model to defend against adversarial attacks. Their defense is similar to Wu et al. [63] except that they are used under different scenarios and their approach does not require modification of the system to be defended. Wu et al. [65] uses a neural vocoder to detect the presence of an adversarial example. Wu et al. [67] uses a "voting" mechanism to detect the presence of an adversarial example. Our work considers a wider variety of ASV models to generate the adversarial attacks while focusing on the susceptibility of *multimodal* ASV models to adversarial attacks — other works have not considered the multimodal scenario.

Adversarial attacks on spoofing countermeasures: The work by Liu et al. [32] evaluated the effect of adversarial attacks on spoofing spoofing CounterMeasures (CMs). They show that the CMs are vulnerable to adversarial attacks in black-box and white-box scenarios. Wu et al. [64] explored defenses against the adversarial attacks that have been shown to work in Liu et al. [32]. They used adversarial training and spatial smoothing as defense mechanisms and show that they can defend against adversarial examples. Wu et al. [63] proposed using a self-supervised pre-trained feature extractor as a defense mechanism. They posited that the pre-trained model would be able to effectively extract the relevant information from the input adversarial example thereby removing the adversarial perturbations. Our work does not explicitly target CMs with adversarial attacks.

Adversarial attacks on multimodal models for tasks other than ASV: The work by Yu et al. [72] study the effect of FGSM and PGD adversarial attacks on multimodal data fusion. They fused information from RGB and thermal infrared images to perform a segmentation task. They found that an attack on either or both of the modalities is a cause for concern due to their vulnerability — they posit that the cause behind the vulnerability is due to the features being non-robust and a lack of proper usage of complementary information from multimodal inputs. Tian and Xu [53] conducted a more in-depth evaluation of adversarial attack on multimodal models in the audio-visual event recognition task and propose a defense mechanism. Their findings are similar to Yu et al. [72] in that the mul-

timodal models are also vulnerable to adversarial attacks. They also find that multimodal adversarial attacks against the models can be more effective than attacks against unimodal models. Evtimov et al. [16] study adversarial attacks against text-image multimodal models under various levels of gray-box assumptions. Compared to the white-box adversary models in Yu et al. [72] and Tian and Xu [53], the work by Evtimov et al. [16] showcases that even under an adversary model with limited information, attacks can be mounted against the multimodal models. Compared to previous works, we conduct adversarial attacks on ASV models.

Adversarial Spoofing Attacks: Gomez-Alanis et al. [18] proposes a network to convert a spoofing sample to an adversarial spoofing sample. They use this to create samples capable of bypassing the CM and ASV mechanisms. Kassis and Hengartner [23] propose a novel attack to generate a spoofing sample capable of bypassing a complete authentication pipeline. Our work does not investigate the concept of bypassing both CM and ASV mechanisms explicitly. While this will be a natural next step, we first have to understand the vulnerability of multimodal ASVs to adversarial attacks since they've not been studied before. In addition, our end goal is to test the feasibility of using the CM as a defense for both adversarial attacks and spoofs against ASVs.

Other audio-based attacks: Vaidya et al. [55] showcased the vulnerability of speech recognition models to a very simple attack mechanism — the perturbed speech can achieve the desired output from the model without alerting the user to the presence of such an attack. Carlini et al. [6] improves Vaidya et al. [55] by generating a more stronger attack and evaluating under stricter conditions. Alzantot et al. [1] used a genetic algorithm combined with a black-box oracle to generate attacks that sound benign to humans but not the speech recognition system. They aimed to create a more stealthy attack compared to Carlini et al. [6]. Hussain et al. [22] used input transformation functions to detect the presence of adversarial examples in speech recognition. Our work does not target the speech recognition pipeline. Chugh et al. [8] proposes a model for detecting multimodal deepfakes using contrastive loss. They work on a dataset that is mainly for video deepfakes but includes an audio voice conversion mechanism of creating audio deepfakes (spoof).

Chapter 8

Conclusion & Future Work

Although we were able to show that augmenting a unimodal spoofing CounterMeasure (CM) with visual information did not introduce a significant run-time overhead, the results in Chapter 4 indicate that this paradigm of augmenting a unimodal CM with visual information did not improve the multimodal CM's EER. A possible future work based on this is through taking a different approach to creating multimodal CMs. For example, a multimodal CM can be created through the use of a contrastive loss like [8].

In Chapter 5, we explored the effect of adversarial attacks on multimodal Automatic Speaker Verifications (ASVs). We identified the high reliance of the multimodal models on the visual modality and showed that this can lead to significantly high attack success rates. We explored two different adversary models and showed that under a more restrictive adversary model, the adversary can mount successful attacks in less than 10% of the examples. Possible future work for this can be by generating stronger attacks that combine adversarial attacks onto spoofs to bypass the spoofing CounterMeasure-Automatic Speaker Verification (CM-ASV) pipeline or attacks targeting the CM alone.

In Chapter 6, we showed that the CM can be extended to detect adversarial attacks targeting ASVs but at a significant cost to the legitimate examples detection rate. We can extend this work by trying additional defense mechanisms like temporal dependency, quantization, downsampling [71] and identifying the number of attacks that the CM can detect but not the defense. A possible future work can be to combine adversarial training as a defense for attacks that target the complete CM-ASV pipeline.

Another promising future avenue that can be explored is as follows: The newer datasets available for CMs development do not provide visual information and therefore experiments

of multimodal CMs cannot be conducted on them. The creation of datasets with newer spoofing attacks along with the visual information is a crucial domain yet to be explored.

References

[1] Moustafa Alzantot, Bharathan Balaji, and Mani Srivastava. Did you hear that? Adversarial Examples Against Automatic Speech Recognition, 2018. arXiv:1801.00554.

[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, et al. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 173–182, 2016.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. arXiv:1409.0473.

[4] Zhongxin Bai and Xiao-Lei Zhang. Speaker recognition based on deep learning: An overview. *Neural networks : the official journal of the International Neural Network Society*, 140:65–99, 2021.

[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines, 2012. arXiv:1206.6389.

[6] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, et al. Hidden Voice Commands. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 513–530, USA, 2016. USENIX Association. ISBN 9781931971324.

[7] Chandramouli Chandrasekaran, Andrea Trubanova, Sébastien Stillittano, et al. The Natural Statistics of Audiovisual Speech. *PLOS Computational Biology*, 5(7):1–18, 07 2009. doi: 10.1371/journal.pcbi.1000436. URL https://doi.org/10.1371/journal.pcbi.1000436.

[8] Komal Chugh, Parul Gupta, Abhinav Dhall, and Ramanathan Subramanian. Not made for each other- Audio-Visual Dissonance-based Deepfake Detection and Localization, 2021. arXiv:2005.14405.

[9] Joon Son Chung and Andrew Zisserman. Out of time: Automated lip sync in the wild. In *ACCV Workshops*, 2016.

[10] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. VoxCeleb2: Deep Speaker Recognition. In *Proc. Interspeech 2018*, pages 1086–1090, 2018. doi: 10.21437/Interspeech.2018-1929.

[11] Rohan Kumar Das, Xiaohai Tian, Tomi Kinnunen, and Haizhou Li. The Attacker's Perspective on Automatic Speaker Verification: An Overview. In *Proc. Interspeech 2020*, pages 4213–4217, 2020. doi: 10.21437/Interspeech.2020-1052.

[12] Najim Dehak, Patrick J. Kenny, Réda Dehak, et al. Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011. doi: 10.1109/TASL.2010.2064307.

[13] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, et al. Boosting Adversarial Attacks with Momentum, 2018. arXiv:1710.06081.

[14] Ariel Ephrat, Inbar Mosseri, Oran Lang, et al. Looking to listen at the cocktail party. *ACM Transactions on Graphics*, 37(4):1–11, Aug 2018. ISSN 1557-7368. doi: 10.1145/3197517.3201357. URL http://dx.doi.org/10.1145/3197517.3201357.

[15] Serife Kucur Ergünay, Elie Khoury, Alexandros Lazaridis, and Sébastien Marcel. On the vulnerability of speaker verification to realistic voice spoofing. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–6, 2015. doi: 10.1109/BTAS.2015.7358783.

[16] Ivan Evtimov, Russel Howes, Brian Dolhansky, et al. Adversarial Evaluation of Multimodal Models under Realistic Gray Box Assumption, 2021. arXiv:2011.12902.

[17] Jianqing Fan, Cong Ma, and Yiqiao Zhong. A selective overview of deep learning. *Statistical science : a review journal of the Institute of Mathematical Statistics*, 36 2: 264–290, 2021.

[18] Alejandro Gomez-Alanis, Jose A. Gonzalez-Lopez, and Antonio M. Peinado. Adversarial Transformation of Spoofing Attacks for Voice Biometrics, 2022. arXiv:2201.01226.

[19] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, 2014. arXiv:1412.6572.

[20] Alex Graves and Navdeep Jaitly. Towards End-to-End Speech Recognition with Recurrent Neural Networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1764–II–1772. JMLR.org, 2014.

[21] Andrew G. Howard, Menglong Zhu, Bo Chen, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017. arXiv:1704.04861.

[22] Shehzeen Hussain, Paarth Neekhara, Shlomo Dubnov, et al. WaveGuard: Understanding and Mitigating Audio Adversarial Examples, 2021. arXiv:2103.03344.

[23] Andre Kassis and Urs Hengartner. Practical Attacks on Voice Spoofing Countermeasures, 2021. arXiv:2107.14642.

[24] Tomi Kinnunen and Haizhou Li. An Overview of Text-Independent Speaker Recognition: From Features to Supervectors. *Speech Commun.*, 52(1):12–40, January 2010. ISSN 0167-6393. doi: 10.1016/j.specom.2009.08.009. URL https://doi.org/10.1016/j.specom.2009.08.009.

[25] Tomi Kinnunen, Md. Sahidullah, Héctor Delgado, et al. The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection. In *Proc. Interspeech 2017*, pages 2–6, 2017. doi: 10.21437/Interspeech.2017-1111.

[26] P. Korshunov, S. Marcel, H. Muckenhirn, et al. Overview of BTAS 2016 speaker anti-spoofing competition. In *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–6, 2016. doi: 10.1109/BTAS.2016.7791200.

[27] Felix Kreuk, Yossi Adi, Moustapha Cisse, and Joseph Keshet. Fooling End-to-end Speaker Verification by Adversarial Examples, 2018. arXiv:1801.03339.

[28] Galina Lavrentyeva, Sergey Novoselov, Andzhukaev Tseren, et al. STC Antispoofing Systems for the ASVspoof2019 Challenge. In *Proc. Interspeech 2019*, pages 1033–1037, 2019. doi: 10.21437/Interspeech.2019-1768.

[29] Y. LeCun, B. Boser, J. S. Denker, et al. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

[30] Xu Li, Na Li, Jinghua Zhong, et al. Investigating Robustness of Adversarial Samples Detection for Automatic Speaker Verification, 2020. arXiv:2006.06186.

[31] Xu Li, Jinghua Zhong, Xixin Wu, et al. Adversarial Attacks on GMM i-vector based Speaker Verification Systems, 2020. arXiv:1911.03078.

[32] Songxiang Liu, Haibin Wu, Hung yi Lee, and Helen Meng. Adversarial Attacks on Spoofing Countermeasures of automatic speaker verification, 2019. arXiv:1910.08716.

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, et al. Towards Deep Learning Models Resistant to Adversarial Attacks, 2019. arXiv:1706.06083.

[34] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models, 2017. arXiv:1708.02182.

[35] Tomas Mikolov, Ilya Sutskever, Kai Chen, et al. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[36] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. VoxCeleb: A Large-Scale Speaker Identification Dataset. In *Proc. Interspeech 2017*, pages 2616–2620, 2017. doi: 10.21437/Interspeech.2017-950.

[37] Arun Narayanan and DeLiang Wang. Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7092–7096, 2013. doi: 10.1109/ICASSP.2013.6639038.

[38] Christopher Olah. Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[39] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training Recurrent Neural Networks, 2013. arXiv:1211.5063.

[40] Alec Radford, Jeff Wu, Rewon Child, et al. Language Models are Unsupervised Multitask Learners. 2019. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[41] Karthik Ramesh, Chao Xing, Wupeng Wang, et al. Vset: A Multimodal Transformer for Visual Speech Enhancement. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6658–6662, 2021. doi: 10.1109/ICASSP39728.2021.9414053.

[42] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

[43] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker Verification Using Adapted Gaussian Mixture Models. *Digit. Signal Process.*, 10:19–41, 2000.

[44] Joseph Roth, Sourish Chaudhuri, Ondrej Klejch, et al. Ava Active Speaker: An Audio-Visual Dataset for Active Speaker Detection. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4492–4496, 2020. doi: 10.1109/ICASSP40776.2020.9053900.

[45] Leda Sarı, Kritika Singh, Jiatong Zhou, et al. A Multi-View Approach To Audio-Visual Speaker Verification, 2021. arXiv:2102.06291.

[46] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015. doi: 10.1109/CVPR.2015.7298682.

[47] Jean-Luc Schwartz and Christophe Savariaux. No, There Is No 150 ms Lead of Visual Speech on Auditory Speech, but a Range of Audiovisual Asynchronies Varying from Small Audio Lead to Large Audio Lag. *PLOS Computational Biology*, 10(7):1–10, 07 2014. doi: 10.1371/journal.pcbi.1003743. URL https://doi.org/10.1371/journal.pcbi.1003743.

[48] Suwon Shon, Tae-Hyun Oh, and James Glass. Noise-tolerant Audio-visual Online Person Verification using an Attention-based Neural Network Fusion, 2018. arXiv:1811.10813.

[49] Suwon Shon, Hao Tang, and James Glass. Frame-level speaker embeddings for text-independent speaker recognition and analysis of end-to-end model, 2018. arXiv:1809.04437.

[50] David Snyder, Daniel Garcia-Romero, Gregory Sell, et al. X-Vectors: Robust DNN Embeddings for Speaker Recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, 2018. doi: 10.1109/ICASSP.2018.8461375.

[51] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, et al. Intriguing properties of neural networks, 2013. arXiv:1312.6199.

[52] Ruijie Tao, Rohan Kumar Das, and Haizhou Li. Audio-visual Speaker Recognition with a Cross-modal Discriminative Network, 2020. arXiv:2008.03894.

[53] Yapeng Tian and Chenliang Xu. Can audio-visual integration strengthen robustness under multimodal attacks?, 2021. arXiv:2104.02000.

[54] Du Tran, Lubomir D. Bourdev, Rob Fergus, et al. C3D: Generic Features for Video Analysis. *ArXiv*, arXiv:1412.0767, 2014.

[55] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. Cocaine Noodles: Exploiting the Gap between Human and Machine Speech Recognition. In *Proceedings of the 9th USENIX Conference on Offensive Technologies*, WOOT'15, page 16, USA, 2015. USENIX Association.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[57] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015. doi: 10.1109/CVPR.2015.7298935.

[58] Wupeng Wang, Chao Xing, Dong Wang, et al. A Robust Audio-Visual Speech Enhancement Model. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7529–7533, 2020. doi: 10.1109/ICASSP40776.2020.9053033.

[59] Xin Wang and Junichi Yamagishi. A Comparative Study on Recent Neural Spoofing Countermeasures for Synthetic Speech Detection. In *Proc. Interspeech 2021*, pages 4259–4263, 2021. doi: 10.21437/Interspeech.2021-702.

[60] Xin Wang, Junichi Yamagishi, Massimiliano Todisco, et al. ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech. *Computer Speech & Language*, 64:101114, 2020. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2020.101114. URL https://www.sciencedirect.com/science/article/pii/S0885230820300474.

[61] Yuxuan Wang, R.J. Skerry-Ryan, Daisy Stanton, et al. Tacotron: Towards End-to-End Speech Synthesis. In *Proc. Interspeech 2017*, pages 4006–4010, 2017. doi: 10.21437/Interspeech.2017-1452.

[62] Yandong Wen, Mahmoud Al Ismail, Weiyang Liu, et al. Disjoint Mapping Network for Cross-modal Matching of Voices and Faces, 2018. arXiv:1807.04836.

[63] Haibin Wu, Andy T. Liu, and Hung yi Lee. Defense for Black-box Attacks on Anti-spoofing Models by Self-Supervised Learning, 2020. arXiv:2006.03214.

[64] Haibin Wu, Songxiang Liu, Helen Meng, and Hung yi Lee. Defense against adversarial attacks on spoofing countermeasures of ASV, 2020. arXiv:2003.03065.

[65] Haibin Wu, Po chun Hsu, Ji Gao, et al. Spotting adversarial samples for speaker verification by neural vocoders, 2021. arXiv:2107.00309.

[66] Haibin Wu, Xu Li, Andy T. Liu, et al. Adversarial defense for automatic speaker verification by cascaded self-supervised learning models, 2021. arXiv:2102.07047.

[67] Haibin Wu, Yang Zhang, Zhiyong Wu, et al. Voting for the right answer: Adversarial defense for speaker verification, 2021. arXiv:2106.07868.

[68] Zhizheng Wu, Tomi Kinnunen, Nicholas Evans, et al. ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge. In *Proc. Interspeech 2015*, pages 2037–2041, 2015. doi: 10.21437/Interspeech.2015-462.

[69] Kelvin Xu, Jimmy Ba, Ryan Kiros, et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, 2016. arXiv:1502.03044.

[70] Weilin Xu, David Evans, and Yanjun Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks, 2017. arXiv:1704.01155.

[71] Zhuolin Yang, Bo Li, Pin-Yu Chen, and Dawn Song. Characterizing audio adversarial examples using temporal dependency. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1g4E3C9t7.

[72] Youngjoon Yu, Hong Joo Lee, Byeong Cheon Kim, et al. Investigating Vulnerability to Adversarial Examples on Multimodal Data Fusion in Deep Learning, 2020. arXiv:2005.10987.

[73] Hossein Zeinali, Lukas Burget, Hossein Sameti, et al. Deep Neural Networks and Hidden Markov Models in i-vector-based Text-Dependent Speaker Verification. In *Proc. The Speaker and Language Recognition Workshop (Odyssey 2016)*, pages 24–30, 2016. doi: 10.21437/Odyssey.2016-4.

APPENDICES

Appendix A

Additional results for Chapter 5

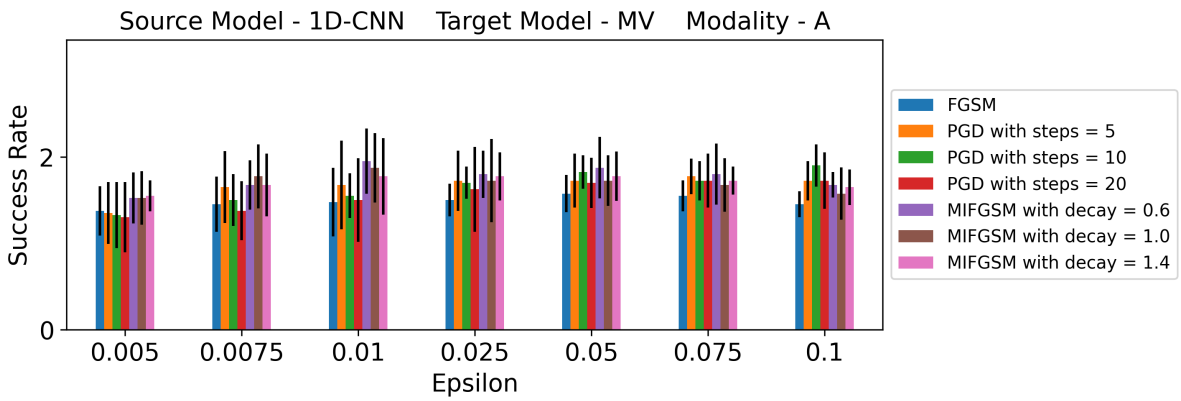


Figure A.1: Figure showing the black-box attack targeting the audio modality by transfer from 1D-CNN to MV for `Read-Only-Template-Access`

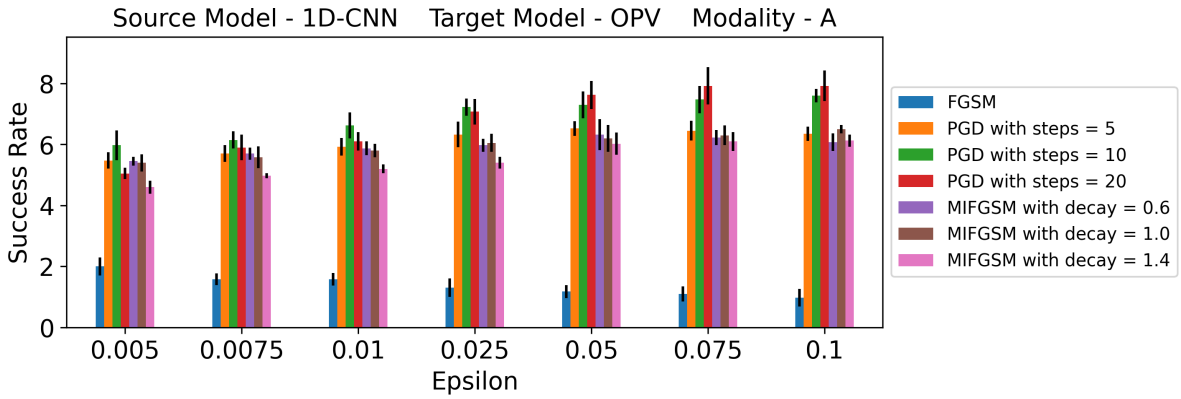


Figure A.2: Figure showing the black-box attack targeting the audio modality by transfer from 1D-CNN to OPV for `Read-Only-Template-Access`

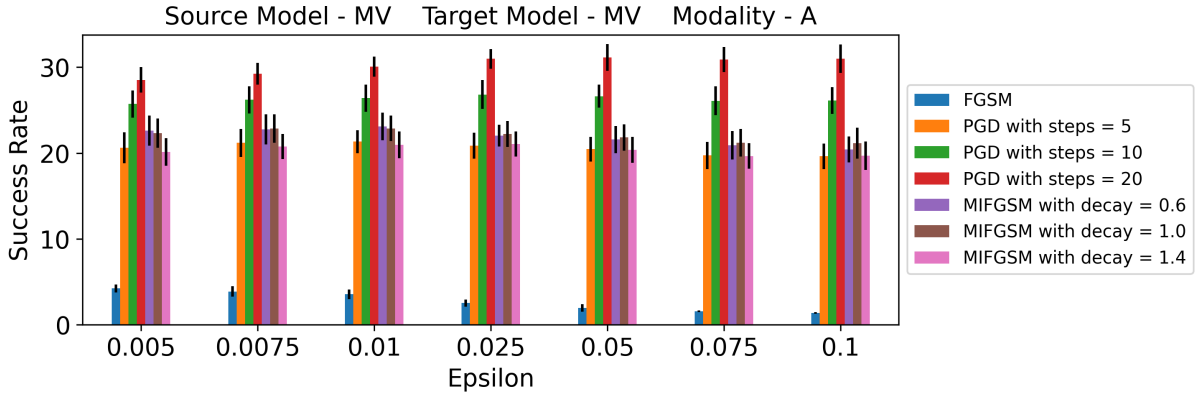


Figure A.3: Figure showing the white-box attack targeting the audio modality on MV for `Read-Only-Template-Access`

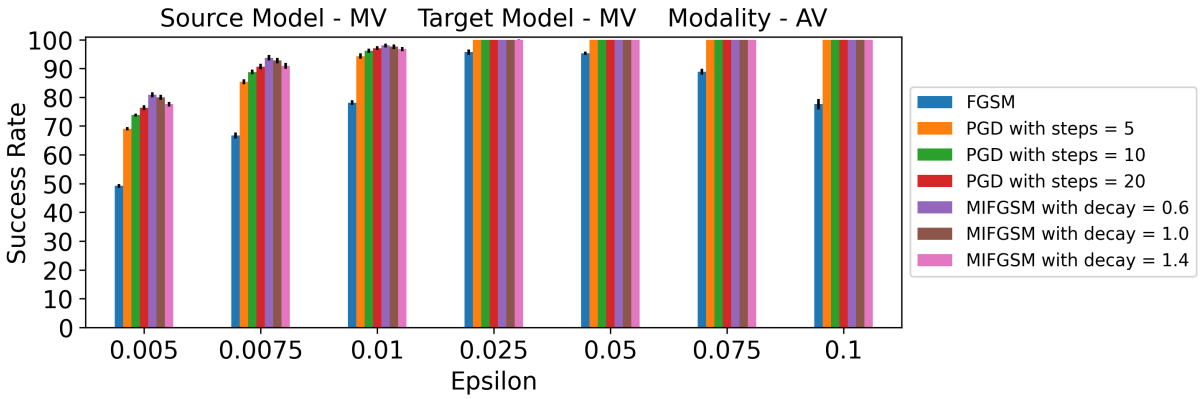


Figure A.4: Figure showing the white-box attack targeting the audio-visual modality on MV for `Read-Only-Template-Access`

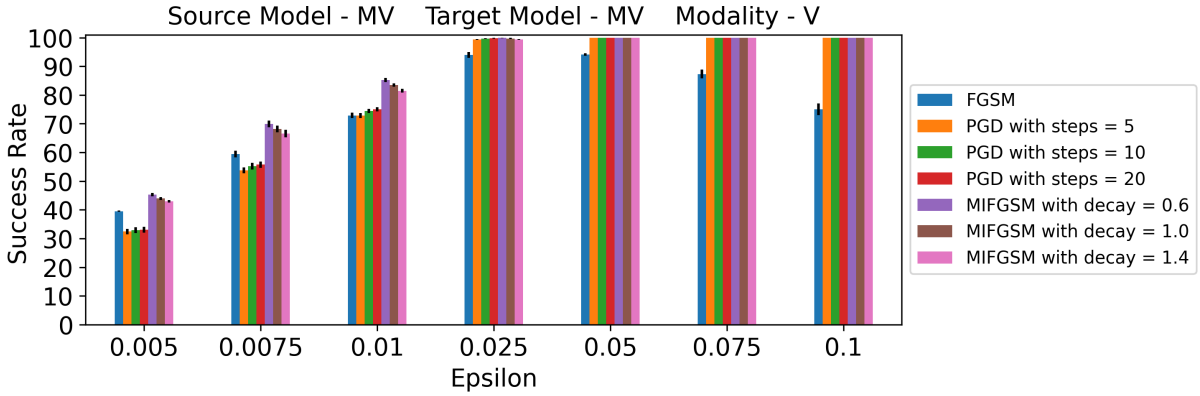


Figure A.5: Figure showing the white-box attack targeting the visual modality on MV for `Read-Only-Template-Access`

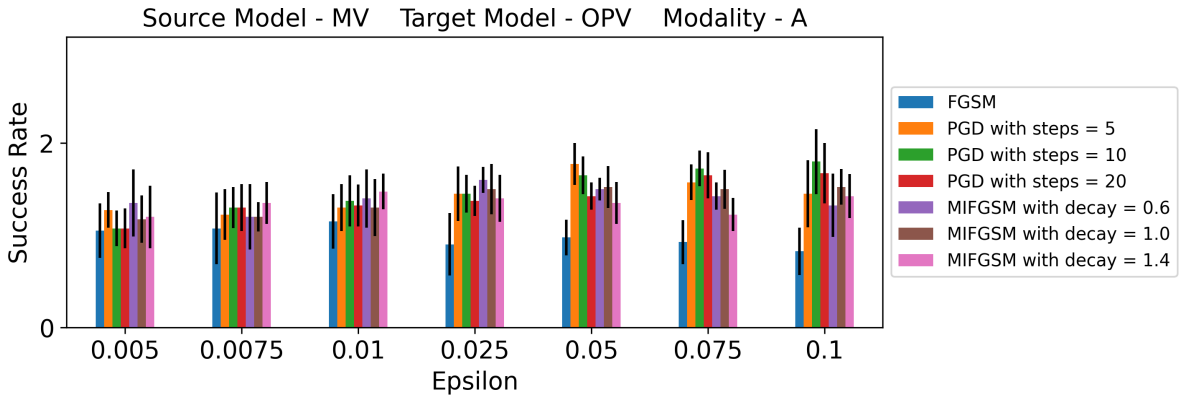


Figure A.6: Figure showing the black-box attack targeting the audio modality by transfer from MV to OPV for `Read-Only-Template-Access`

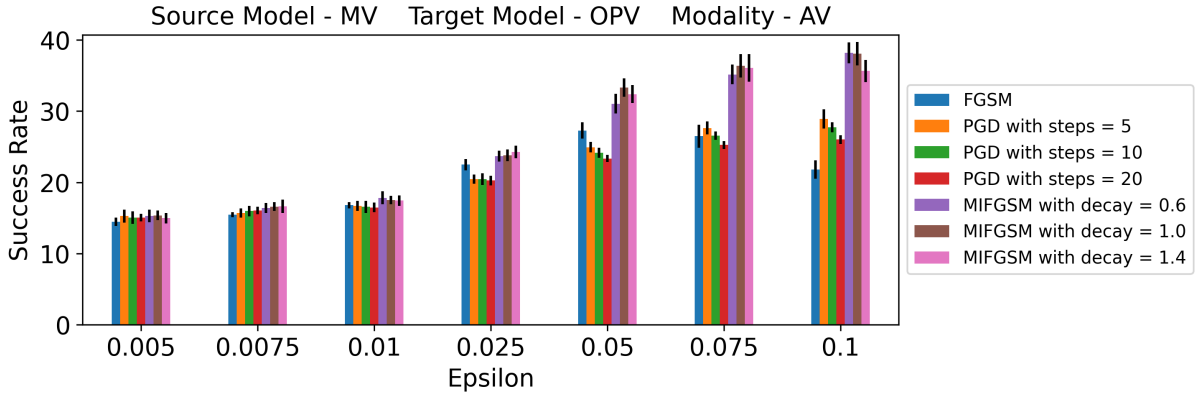


Figure A.7: Figure showing the black-box attack targeting the audio-visual modality by transfer from MV to OPV for `Read-Only-Template-Access`

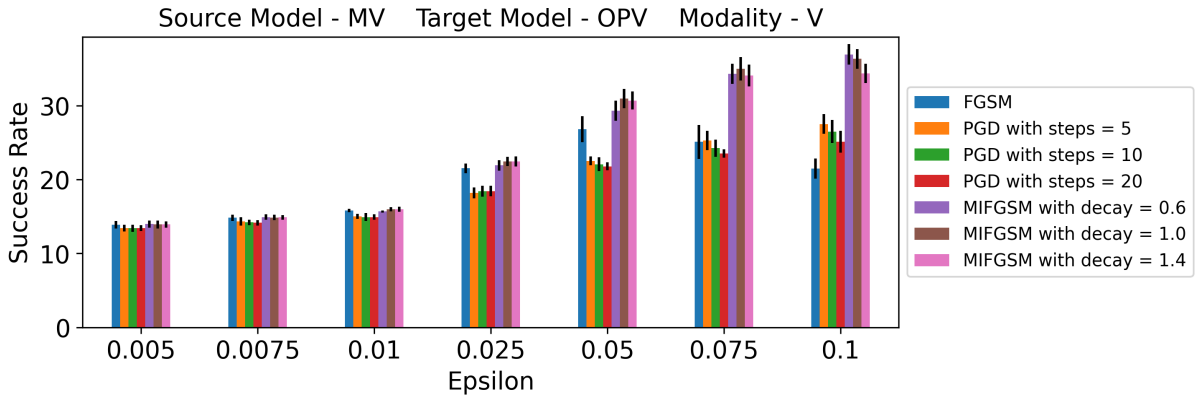


Figure A.8: Figure showing the black-box attack targeting the visual modality by transfer from MV to OPV for `Read-Only-Template-Access`

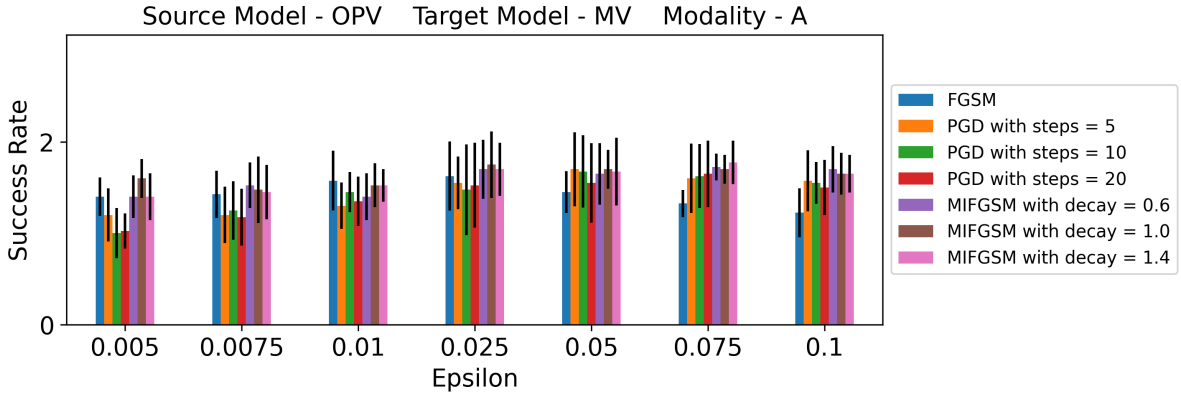


Figure A.9: Figure showing the black-box attack targeting the audio modality by transfer from OPV to MV for `Read-Only-Template-Access`

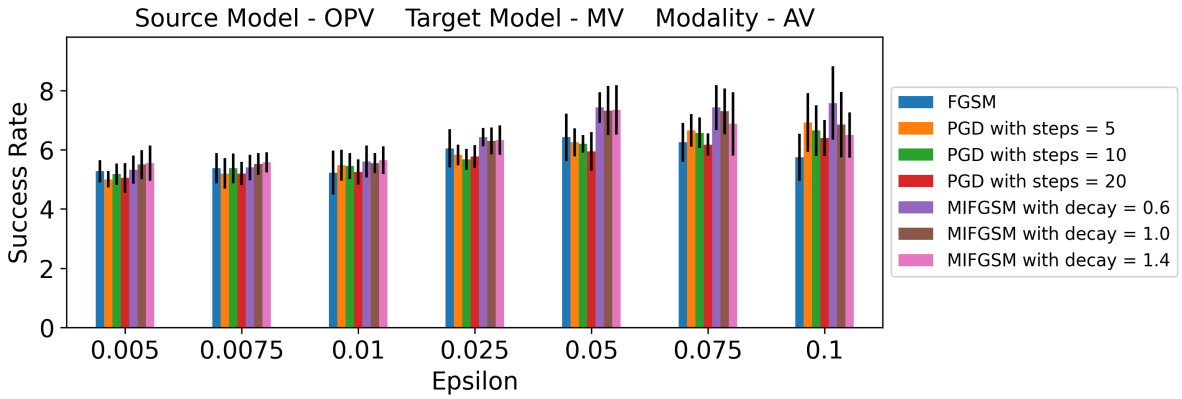


Figure A.10: Figure showing the black-box attack targeting the audio-visual modality by transfer from OPV to MV for `Read-Only-Template-Access`

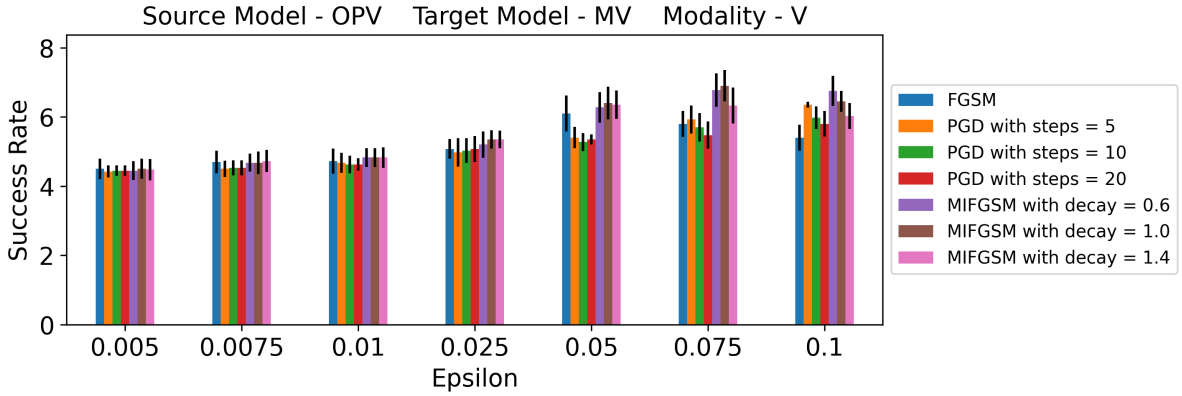


Figure A.11: Figure showing the black-box attack targeting the visual modality by transfer from OPV to MV for `Read-Only-Template-Access`

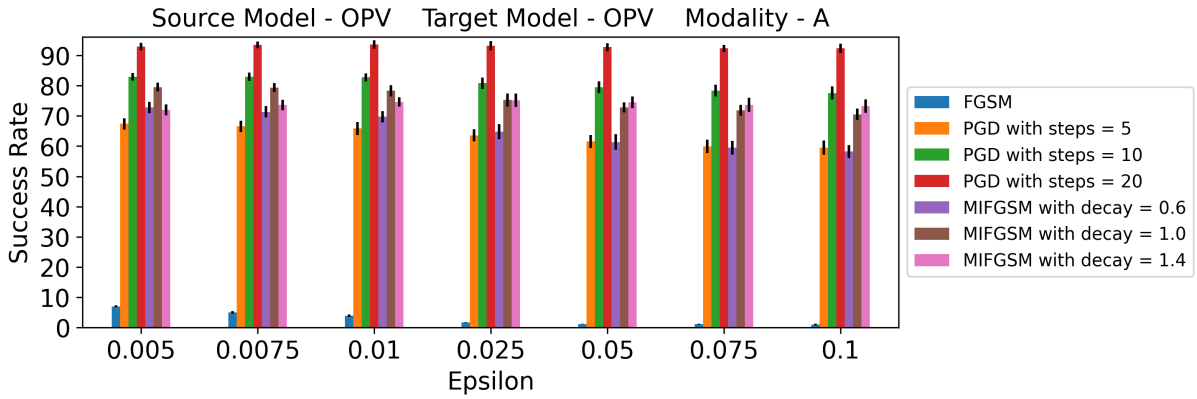


Figure A.12: Figure showing the white-box attack targeting the audio modality on OPV for `Read-Only-Template-Access`

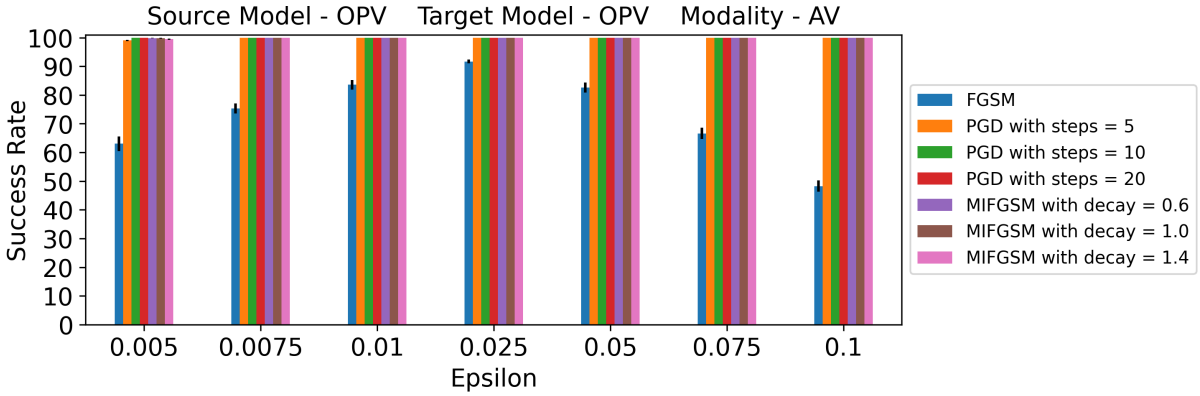


Figure A.13: Figure showing the white-box attack targeting the audio-visual modality on OPV for `Read-Only-Template-Access`

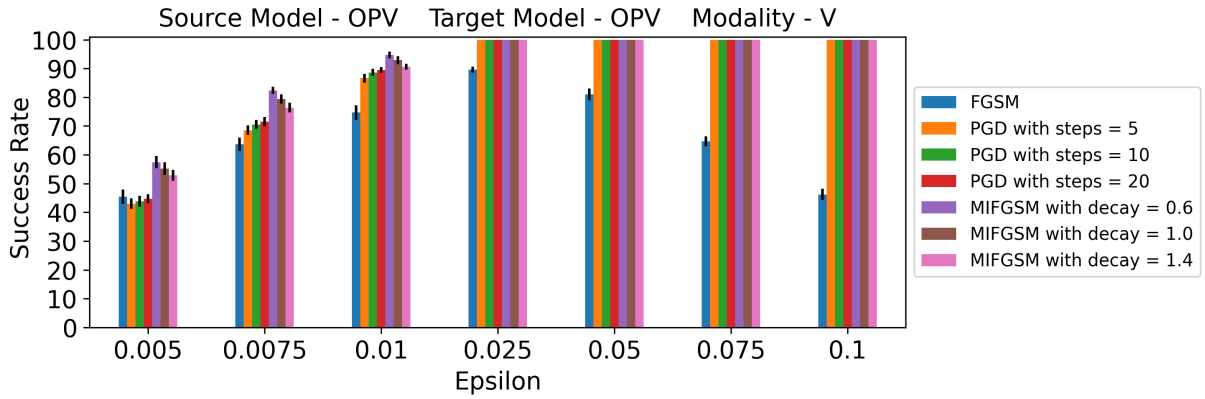


Figure A.14: Figure showing the white-box attack targeting the visual modality on OPV for `Read-Only-Template-Access`

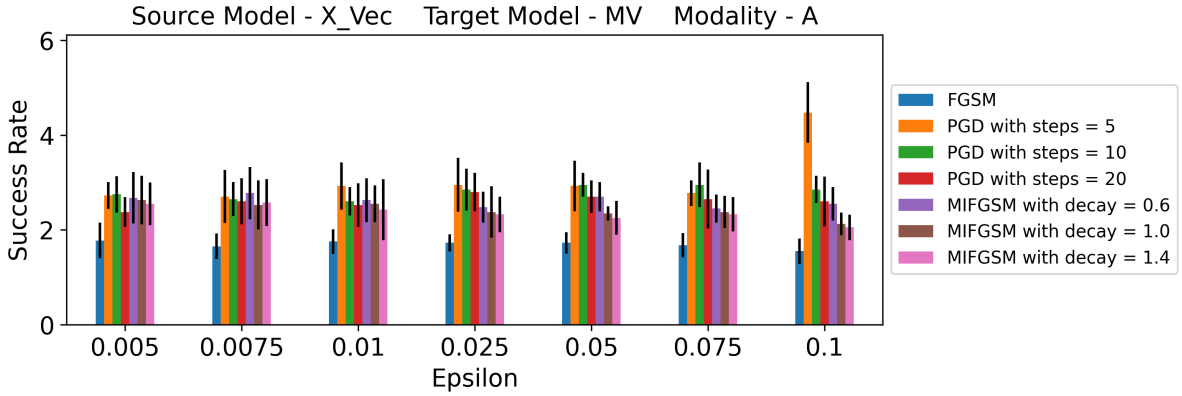


Figure A.15: Figure showing the black-box attack targeting the audio modality by transfer from X-Vec to MV for `Read-Only-Template-Access`

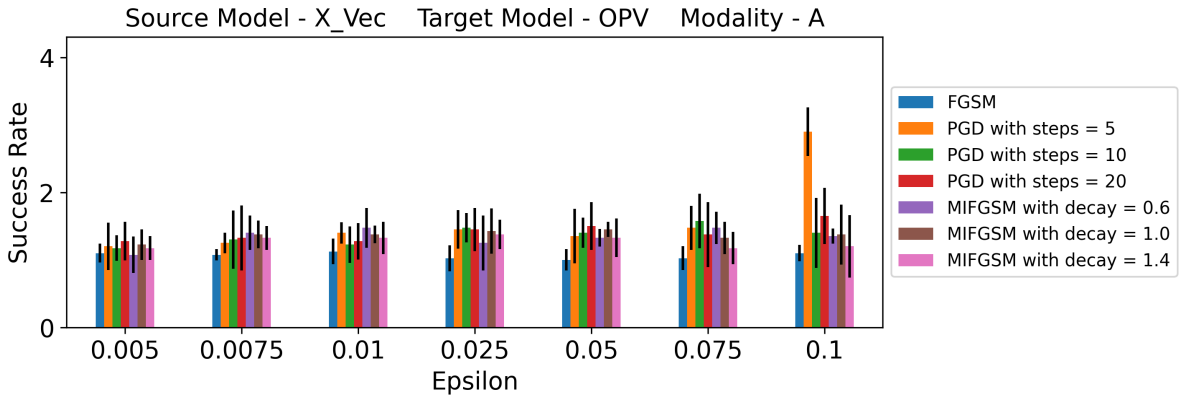


Figure A.16: Figure showing the black-box attack targeting the audio modality by transfer from X-Vec to OPV for `Read-Only-Template-Access`

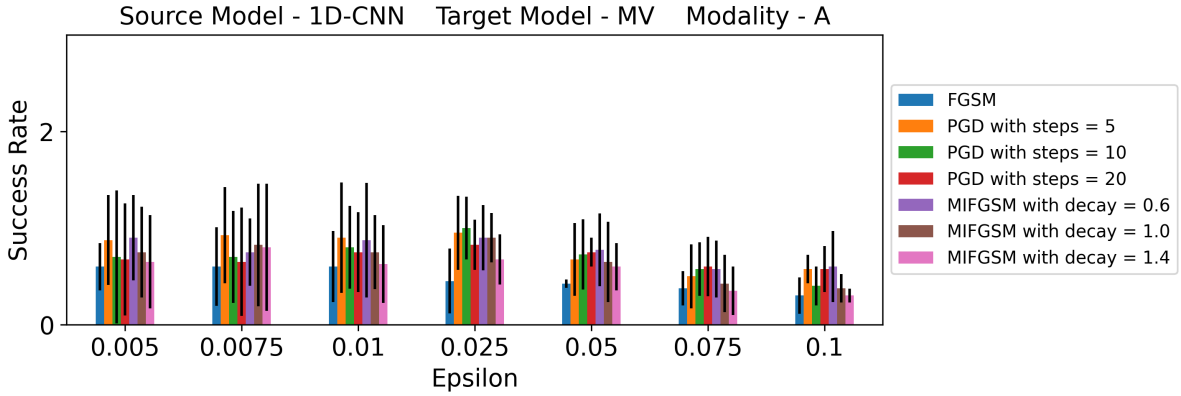


Figure A.17: Figure showing the black-box attack targeting the audio modality by transfer from 1D-CNN to MV for `No-Template-Access`

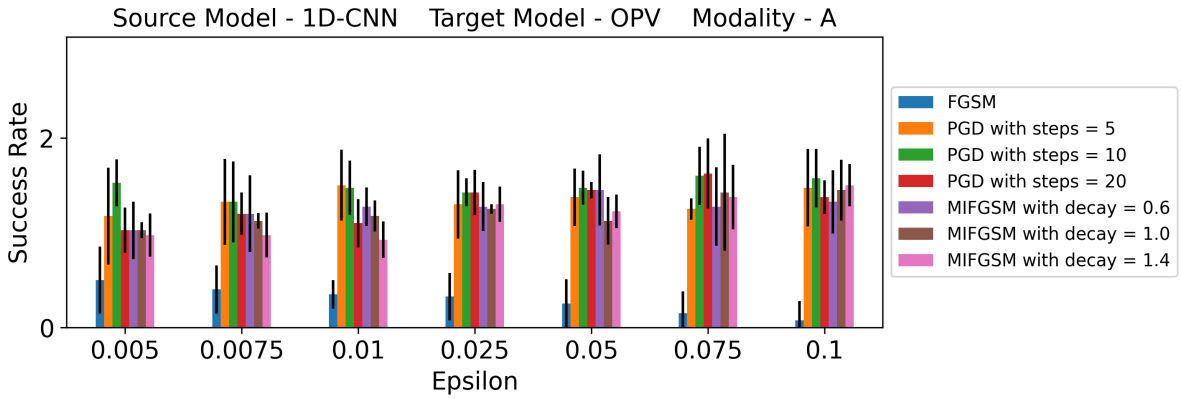


Figure A.18: Figure showing the black-box attack targeting the audio modality by transfer from 1D-CNN to OPV for No-Template-Access

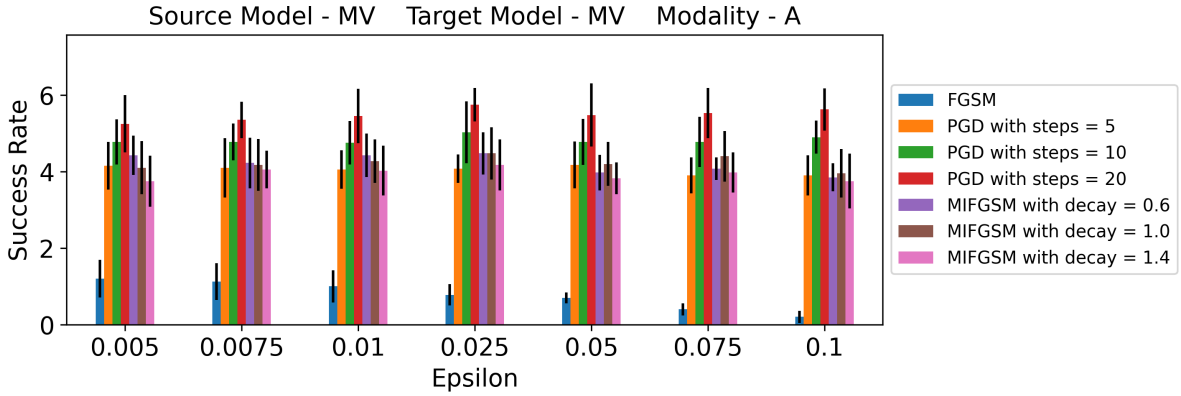


Figure A.19: Figure showing the white-box attack targeting the audio modality on MV for No-Template-Access

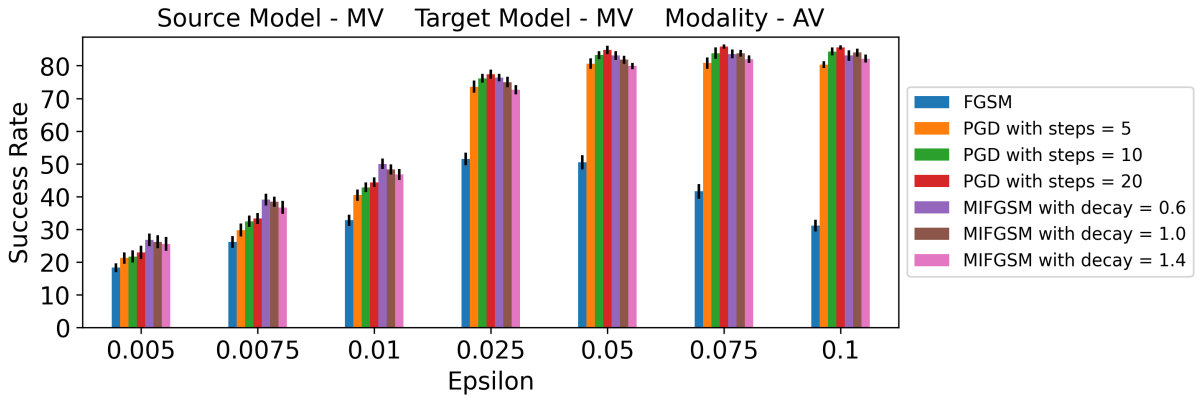


Figure A.20: Figure showing the white-box attack targeting the audio-visual modality on MV for No-Template-Access

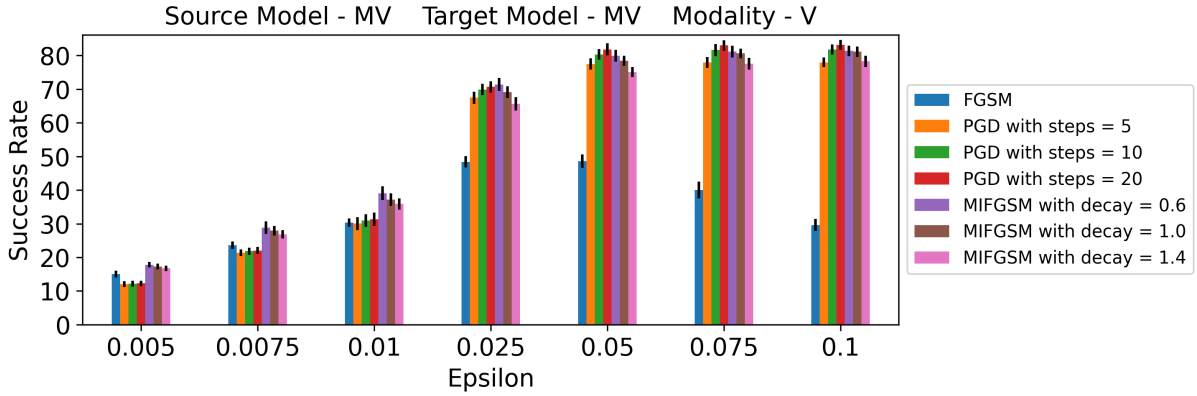


Figure A.21: Figure showing the white-box attack targeting the visual modality on MV for No-Template-Access

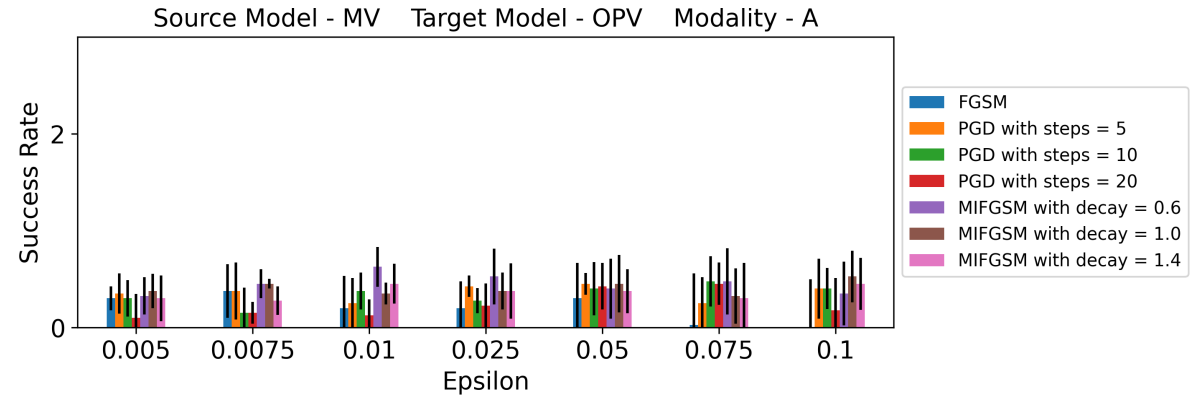


Figure A.22: Figure showing the black-box attack targeting the audio modality by transfer from MV to OPV for `No-Template-Access`

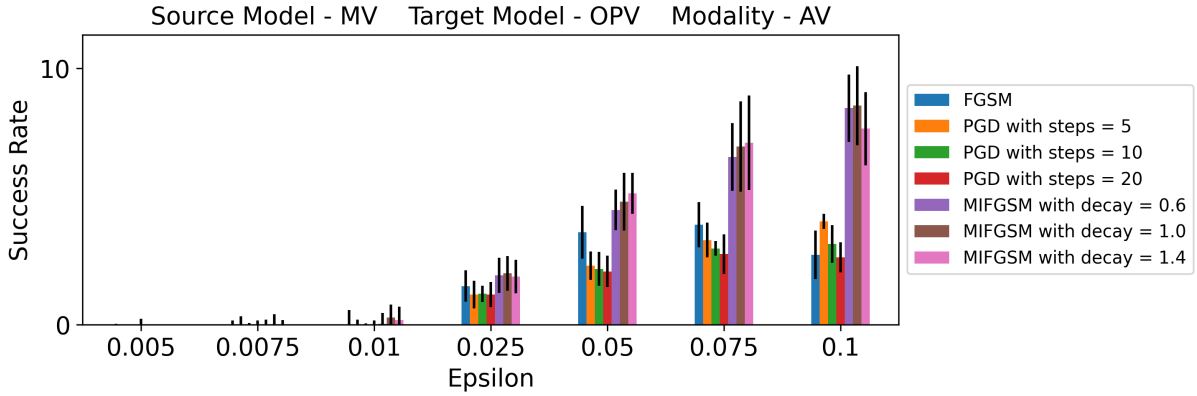


Figure A.23: Figure showing the black-box attack targeting the audio-visual modality by transfer from MV to OPV for `No-Template-Access`

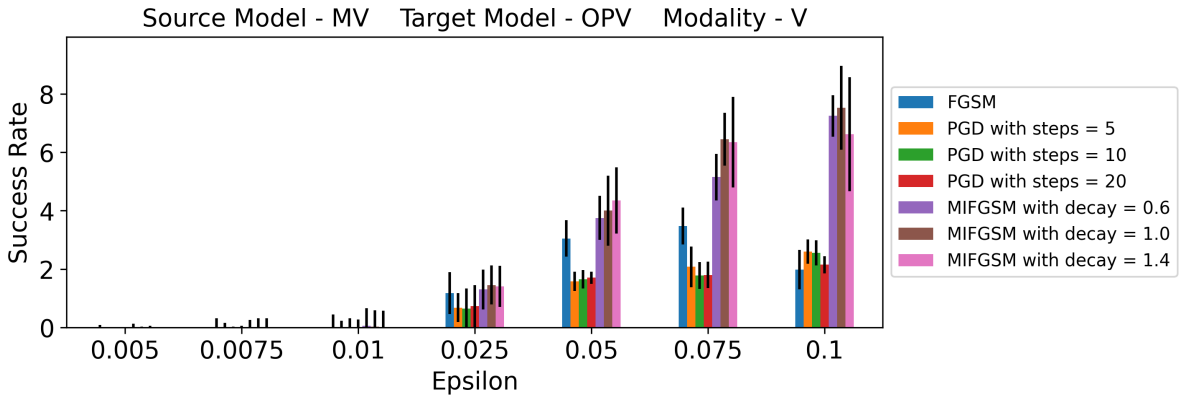


Figure A.24: Figure showing the black-box attack targeting the visual modality by transfer from MV to OPV for `No-Template-Access`

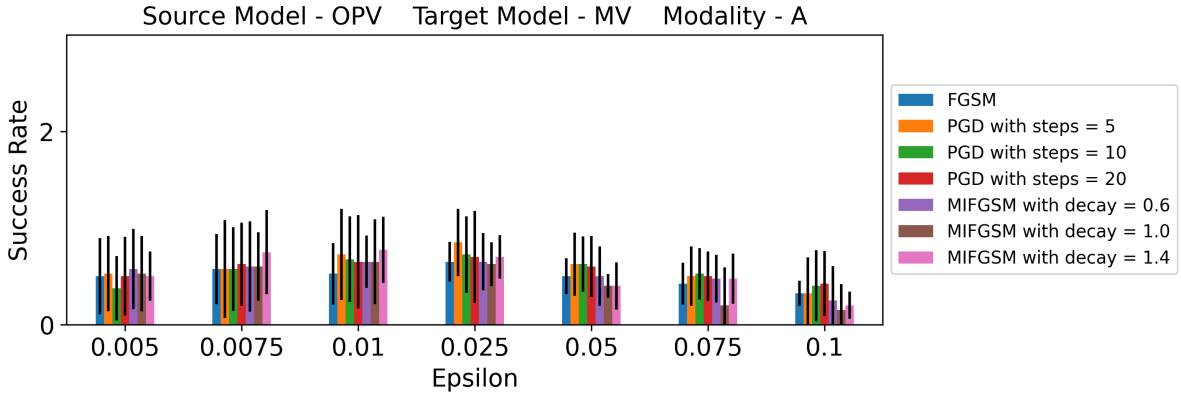


Figure A.25: Figure showing the black-box attack targeting the audio modality by transfer from OPV to MV for `No-Template-Access`

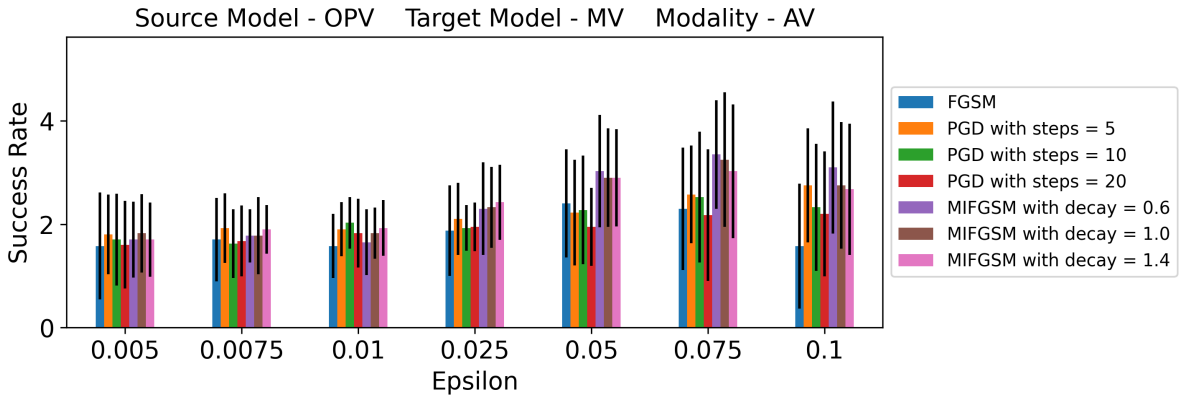


Figure A.26: Figure showing the black-box attack targeting the audio-visual modality by transfer from OPV to MV for `No-Template-Access`

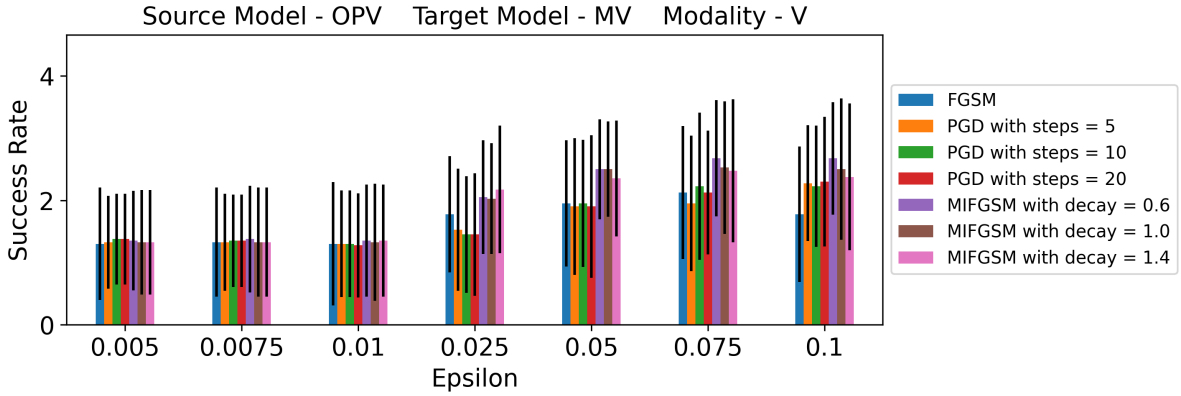


Figure A.27: Figure showing the black-box attack targeting the visual modality by transfer from OPV to MV for `No-Template-Access`

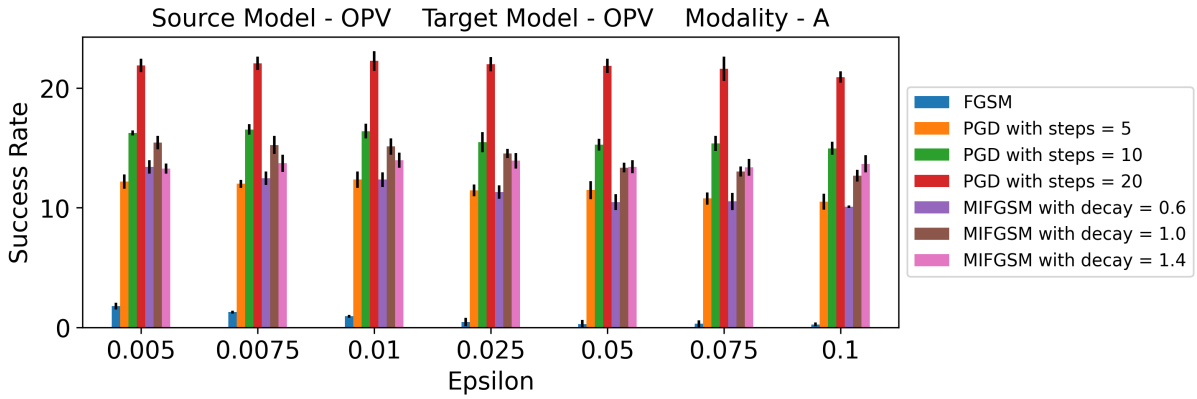


Figure A.28: Figure showing the white-box attack targeting the audio modality on OPV for No-Template-Access

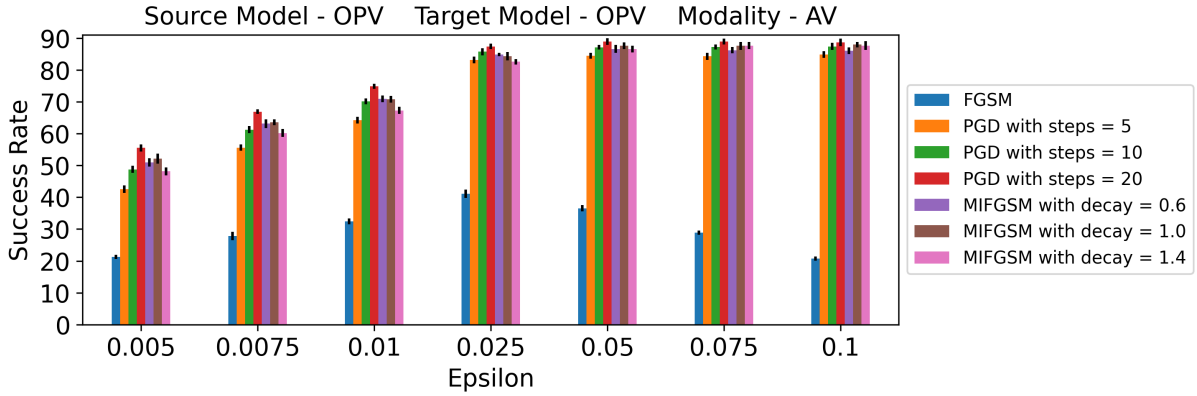


Figure A.29: Figure showing the white-box attack targeting the audio-visual modality on OPV for No-Template-Access

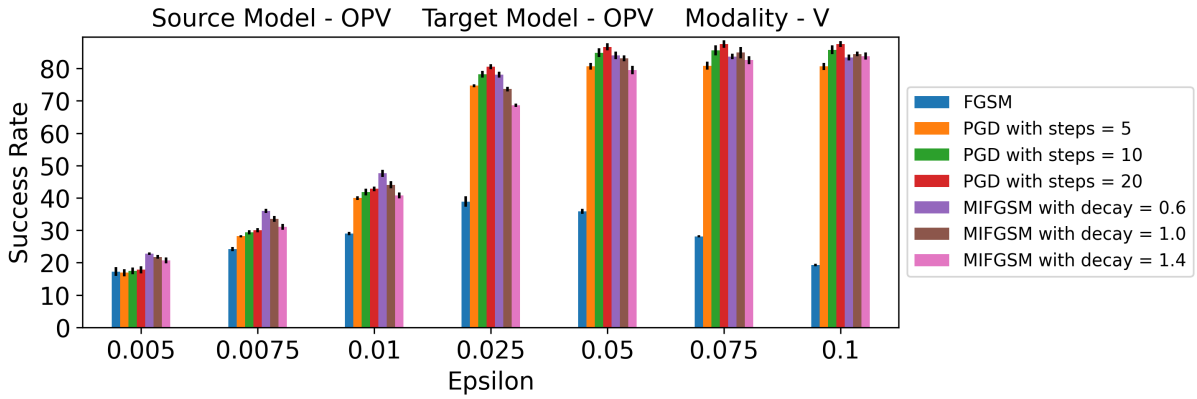


Figure A.30: Figure showing the white-box attack targeting the visual modality on OPV for No-Template-Access

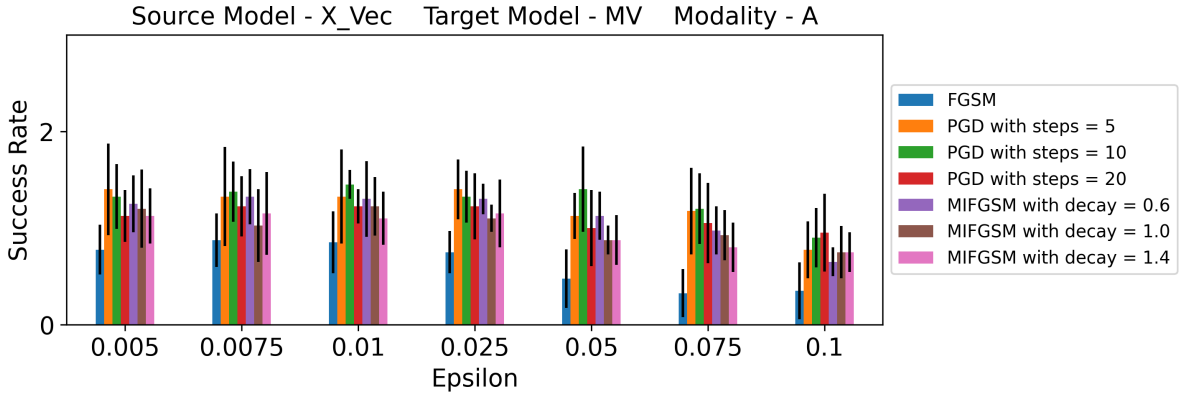


Figure A.31: Figure showing the black-box attack targeting the audio modality by transfer from X-Vec to MV for No-Template-Access

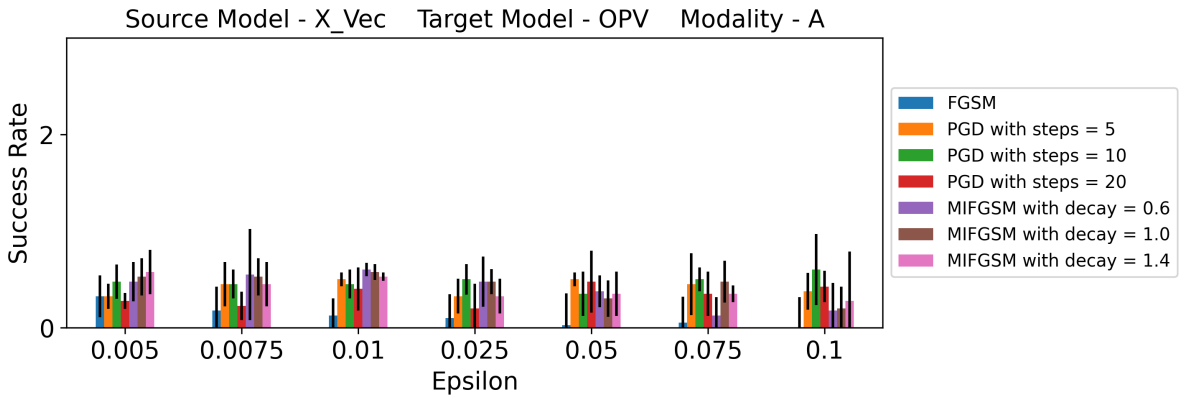


Figure A.32: Figure showing the black-box attack targeting the audio modality by transfer from X-Vec to OPV for `No-Template-Access`