



**UNIVERSIDAD
DE GRANADA**

**TRABAJO FIN DE GRADO
INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN**

**Evaluación sobre la aplicabilidad de los
actuales conversores y sintetizadores de
voz para la generación de voz falsificada**

Autor

Francisco López Houdaibi

Directores

Ángel Manuel Gómez García
Antonio Miguel Peinado
Herreros



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS
INFORMÁTICA Y DE TELECOMUNICACIÓN**

Granada, septiembre de 2024



UNIVERSIDAD
DE GRANADA

Evaluación sobre la aplicabilidad de los actuales conversores y sintetizadores de voz para la generación de voz falsificada

Autor

Francisco López Houdaibi

Directores

Ángel Manuel Gómez García
Antonio Miguel Peinado
Herreros



DEPARTAMENTO DE TEORÍA DE LA SEÑAL, TELEMÁTICA Y COMUNICACIONES
Granada, septiembre de 2024

Evaluación sobre la aplicabilidad de los actuales conversores y sintetizadores de voz para la generación de voz falsificada

Francisco López Houdaibi

Palabras clave: Locutor objetivo, Locutor objetivo sintetizado, Locutor impostor, EER, ASV, Spoofing, IA, TTS.

Resumen

En este trabajo se presenta la evaluación del rendimiento de diferentes sistemas de verificación automática del locutor (ASV) basados en redes neuronales. Se genera la síntesis de voz de múltiples locutores inscritos en el sistema de ASV mediante sistemas de síntesis de texto a voz (TTS) basados en redes neuronales. De esta manera, se realizan ataques de suplantación de identidad en los diferentes sistemas de ASV para evaluar el rendimiento práctico de dichos sistemas, utilizando para ello, una métrica del rendimiento del sistema de ASV denominada tasa de error equivalente (EER).

El trabajo aborda, en primer lugar, la introducción y motivación, destacando la importancia de la inteligencia artificial (IA) y los peligros que pueden implicar las tecnologías derivadas de dicho campo, por ejemplo, en la suplantación de identidad en ciertos sistemas biométricos como el reconocimiento del locutor. Posteriormente, se realiza un estudio teórico de las diversos sistemas existentes en el estado del arte, los cuales son utilizados en la fase experimental. Estos sistemas incluyen ASV basados en arquitecturas codificadoras como redes neuronales de retardo temporal (TDNN), redes residuales (ResNet) y TDNN con atención, propagación y agregación de canales enfatizados (ECAPA-TDNN), así como sistemas de TTS como FastSpeech, basado en Transformer, y Tacotron, con arquitectura de red neuronal recurrente (RNN); y sintetizadores de voz (vocoders) como DiffWave, basados en redes probabilísticas, y HiFi-GAN, basado en redes adversarias generativas. Finalmente, todos estos modelos, los cuales están pre-entrenados en ciertas bases de datos, se utilizan para realizar los diferentes experimentos de evaluación de modelos de síntesis y verificación de locutor basados en IA.

Se incluyen diversos resultados experimentales como histogramas de puntuaciones de locutores objetivos, impostores y objetivos sintetizados y curvas de compensación de errores de detección (DET), muy útiles para comparar el rendimiento de los diferentes sistemas de ASV. A partir de las diferentes gráficas y tablas obtenidas en los experimentos, se evidencia la vulnerabilidad de los sistemas de ASV frente a este tipo de ataques, conocidos también como ataques de spoofing.

Evaluation of the applicability of current voice converters and synthesizers for fake voice generation

Francisco López Houdaibi

Keywords: Target speaker, Synthesized target speaker, Impostor speaker, EER, ASV, Spoofing, AI, TTS.

Abstract

This work presents an evaluation of the performance of various automatic speaker verification (ASV) systems based on neural networks. Voice synthesis of multiple speakers enrolled in the ASV system is generated using neural network-based text-to-speech (TTS) systems. In this way, identity spoofing attacks are carried out on different ASV systems to evaluate the practical performance of these systems, using a performance metric for ASV systems known as Equal Error Rate (EER).

The paper first addresses the introduction and motivation, highlighting the importance of artificial intelligence (AI) and the potential dangers associated with technologies derived from this field, such as identity spoofing in certain biometric systems like speaker recognition. It then provides a theoretical study of the various existing systems in the state of the art, which are used in the experimental phase. These systems include ASV systems based on encoder architectures such as Time-Delay Neural Networks (TDNN), Residual Networks (ResNet), and TDNN with Attention, Emphasized Channel Propagation and Aggregation (ECAPA-TDNN), as well as TTS systems like FastSpeech, based on Transformer, and Tacotron, with Recurrent Neural Network (RNN) architecture; and voice synthesizers (vocoders) like DiffWave, based on probabilistic networks, and HiFi-GAN, based on Generative Adversarial Networks. Finally, all these models, which are pre-trained on certain databases, are used to perform various experiments for evaluating AI-based synthesis and speaker verification models.

Various experimental results are included, such as histograms of scores for target speakers, impostors, and synthesized targets, as well as Detection Error Tradeoff (DET) curves, which are very useful for comparing the performance of different ASV systems. From the different graphs and tables obtained in the experiments, the vulnerability of ASV systems to this type of attacks, also known as spoofing attacks, is evident.

Yo, **Francisco López Houdaibi** de la titulación Ingeniería de Técnologías de la telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 20077756K, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen. Así mismo, asumo la originalidad del trabajo, entendida en el sentido de que no se han utilizado fuentes sin citarlas debidamente.

Fdo: Francisco López Houdaibi

Granada, a 3 de septiembre
de 2024

Índice general

1	Introducción, motivación, objetivos y estructura del documento	22
1.1	Introducción y motivación	22
1.2	Objetivos	24
1.3	Estructura del documento	24
2	Fundamento Teórico	27
2.1	Introducción	27
2.2	Reconocimiento del locutor	27
2.2.1	Identificación del locutor	29
2.2.2	Verificación del locutor	29
	2.2.2.1. Fases del proceso de verificación del locutor	30
	2.2.2.2. Verificación del locutor contra ataques de suplantación de identidad (spoofing)	31
2.2.3	Extractores de características del locutor	32
	2.2.3.1. x-vector	33
	2.2.3.2. ResNet	35
	2.2.3.3. ECAPA-TDNN	37
2.2.4	Protocolo de evaluación del sistema de ASV frente a ataques de spoofing.	38
	2.2.4.1. Diseño del conjunto de datos	38
	2.2.4.2. Medidas de rendimiento y presentación de resultados <i>Precisión de la verificación</i>	39
		39
	<i>Eficiencia del sistema</i>	42
		42
	<i>Usabilidad del sistema</i>	42
		42
2.3	Síntesis de texto a voz (TTS)	43
2.3.1	Tacotron 2	44
	2.3.1.1. Espectrograma Mel	45
	2.3.1.2. Arquitectura del modelo (Red de predicción del espectrograma)	45
	<i>Codificador de caracteres</i>	46
		46
	<i>Red de atención sensible a la ubicación</i>	46
		46
	<i>Decodificador de espectrograma Mel</i>	47
2.3.2	FastSpeech 2	49

2.3.2.1.	Embedder del fonema	50
2.3.2.2.	Codificador y decodificador del espectrograma de Mel	50
2.3.2.3.	Adaptador de varianza	50
	<i>Predictor de duración</i>	
	· · · · ·	51
	<i>Predictor de pitch</i>	
	· · · · ·	51
	<i>Predictor de energía</i>	
	· · · · ·	52
2.3.2.4.	Transformer	53
	<i>Motivación para el Desarrollo del Transformer</i>	
	· · · · ·	53
	<i>Arquitectura del codificador</i>	
	· · · · ·	53
	<i>Arquitectura del decodificador</i>	
	· · · · ·	58
2.3.3	Vocoder: HiFi-GAN	60
2.3.3.1.	Generador	60
2.3.3.2.	Discriminador	61
2.3.3.3.	Funciones de pérdida en el entrenamiento	62
	<i>Pérdida GAN</i>	
	· · · · ·	62
	<i>Pérdida del espectrograma de Mel</i>	
	· · · · ·	62
	<i>Pérdida de emparejamiento de características</i>	
	· · · · ·	63
2.3.4	Vocoder: DiffWave	63
2.3.4.1.	Modelado de la densidad de probabilidad para la síntesis de datos	64
2.3.4.2.	Arquitectura del modelo de difusión	64
3	Estudio experimental de los sistemas TTS y ASV analizados teóricamente	68
3.1	Introducción	68
3.2	Estudio del rendimiento de los sistemas ASV utilizados	68
3.2.1	Caracterización del locutor objetivo	69
3.2.2	Caracterización del locutor impostor	70
3.2.3	Métodos	70
	3.2.3.1. Histograma de puntuaciones del locutor objetivo agradado	71
	3.2.3.2. Histograma de puntuaciones del locutor impostor agradado	71
	3.2.3.3. Histograma de puntuaciones del locutor objetivo segregado por sexo	71
	3.2.3.4. Histograma de puntuaciones del locutor impostor segregado por sexo	72
	3.2.3.5. Histograma de puntuaciones conjunto del LO, LI y la EER	72

3.2.3.6. Histograma de puntuaciones conjunto del LO, LI segregado por sexo y la EER	73
3.2.4 Resultados	73
3.2.4.1. Comparación de la EER obtenida con la EER proporcionada por el repositorio	76
3.3 Estudio del rendimiento de los sistemas de ASV utilizados frente a ataques de spoofing	78
3.3.1 Síntesis de voz de la base de datos LJSpeech utilizando FastSpeech2 y DiffWave	79
3.3.1.1. Métodos	79
<i>Síntesis de voz:</i>	79
<i>Histograma de puntuaciones del locutor objetivo:</i>	80
<i>Histograma de puntuaciones del locutor objetivo sintetizado:</i>	80
<i>Histograma de puntuaciones conjunto del LO, LOS y la EER:</i>	80
<i>Inscripción del locutor objetivo:</i>	80
3.3.1.2. Resultados	81
3.3.2 Síntesis de voz de la base de datos LibriTTS utilizando Tacotron2 y HiFiGAN	84
3.3.2.1. Métodos	85
<i>Síntesis de voz:</i>	85
<i>Histograma de puntuaciones del locutor objetivo sintetizado:</i>	85
<i>Histograma de puntuaciones del locutor objetivo:</i>	85
<i>Histograma de puntuaciones conjunto del LOS, LO segregado por sexo masculino:</i>	85
<i>Histograma de puntuaciones conjunto del LOS, LO segregado por sexo femenino:</i>	86
<i>Histograma de puntuaciones conjunto del LOS, LO agragado:</i>	86
3.3.2.2. Resultados	86
3.3.3 Resultado final	88
4 Conclusiones	93

Índice de figuras

2.1	Esquema de identificación de locutor. Imagen adaptada de [47]	28
2.2	Esquema de identificación y verificación del locutor. Imagen adaptada de [47]	28
2.3	Arquitectura del sistema x-vector. Imagen adaptada de [53]	34
2.4	Bloque residual convolucional 2D utilizado como bloque básico en ResNet. Figura adaptada de [66].	36
2.5	Arquitectura del sistema ECAPA-TDNN. Figura adaptada de [4].	38
2.6	Ilustración del marco general utilizado en estudios de ataques de suplantación de identidad. Figura adaptada de [73]	39
2.7	Relación existente entre FRR,FAR y EER	40
2.8	Comparación de las curvas ROC y DET utilizando diferentes algoritmos de clasificación	42
2.9	Funciones de densidad de probabilidad de las puntuaciones ‘scores’ de los locutores legítimos e impostores	43
2.10	Diagrama de bloques general del sistema TTS	44
2.11	Arquitectura de Tacotron 2	48
2.12	Arquitectura general de FastSpeech2	50
2.13	Arquitectura del adaptador de varianza	51
2.14	Esquema detallado del predictor del pitch.	52
2.15	Codificador del Transformer	54
2.16	Decodificador del Transformer	59
2.17	Arquitectura del modelo HiFi-GAN	61
2.18	Arquitectura del modelo de difusión. Proceso Forward (parte superior) y proceso Reverse (parte inferior)	65
3.1	Histogramas de puntuaciones LI, LO (segregado por sexo: Masculino) para diferentes modelos ASV	74
3.2	Tasa de falsos positivos (FPR) y Tasa de falsos negativos (FNR) en función del umbral para diferentes modelos ASV (segregado por sexo: Masculino)	75
3.3	Histogramas de puntuaciones LI vs LO (segregado por sexo: Femenino) para diferentes modelos ASV	75
3.4	Tasa de falsos positivos (FPR) y Tasa de falsos negativos (FNR) en función del umbral para diferentes modelos ASV (segregado por sexo: Femenino)	75
3.5	Histogramas de puntuaciones LI, LO (Gráfica agregada) para diferentes modelos ASV	75
3.6	Tasa de falsos positivos (FPR) y Tasa de falsos negativos (FNR) en función del umbral para diferentes modelos ASV (Gráfica agregada)	76

3.7	Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs (umbral). EP-1000 sentencias	83
3.8	Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs (umbral). EP-100 sentencias	84
3.9	Comparación de las curvas DET para diferentes inscripciones del LO utilizando síntesis del LO mediante FastSpeech2 y Diffwave.	84
3.10	Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs umbral (Segregado por sexo masculino).	88
3.11	Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs umbral (Segregado por sexo masculino).	88
3.12	Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs umbral (Agregado).	89
3.13	Curvas DET de los 3 sistemas de ASV analizados, cuando se realiza síntesis de voz utilizando Tacotron 2 + Diffwave.	89
3.14	Tabla asociada a las tasas de error equivalente (EERs) de los diferentes experimentos realizados.	90

Índice de tablas

2.1	Matriz de confusión	40
2.2	Patrón de atención del ejemplo propuesto	56
3.1	Locutores Objetivos Masculinos de la base de datos LibriTTS . . .	70
3.2	Locutores Objetivos Femenino de la base de datos LibriTTS . . .	70
3.3	EER obtenida en los sistemas ASV estudiados,(<i>veri_test2.txt</i>) .	77
3.4	EER obtenida en los sistemas ASV estudiados,(Resultados propios)	77
3.5	Comparación de la EER estimada sin técnicas de Spoofing vs. con suplantación de identidad	82
3.6	Comparación de la EER estimado con embedding Prototipo (EP) compuesto por 1000 Sentencias vs. 100 Sentencias	82
3.7	Comparación de la EER estimada, para los diferentes sistemas de ASV, en función del sexo	86
3.8	Comparación de la EER de los sistemas de ASV: x-vector, ResNet y ECAPA-TDNN, para la suplantación de locutor mediante dos sistemas de TTS: FastSpeech 2 + DiffWave, utilizando 20 locutores objetivos sintetizados, y Tacotron 2 + HiFi-GAN, utilizando 1 locutor objetivo sintetizado.	91

Listado de Siglas

- IA** Inteligencia Artificial (Artificial Intelligence)
GMM Modelo de Mezcla Gaussiana (Gaussian Mixture Model)
ReLU Unidades Lineales Rectificadas (Rectified Linear Units)
STFT Transformada de Fourier de Tiempo reducido (Short-Time Fourier Transform)
I-STFT Transformada Inversa de Fourier de Tiempo reducido (Inverse Short-Time Fourier Transform)
FFT Alimentación Directa del Transformer (Feed-Forward Transformer) / Transformada Rápida de Fourier (Fast Fourier Transform)
FFN Red de Alimentación Directa (Feed-Forward Network)
MSE Error Cuadrático Mínimo (Minimum Square Error)
MFA Alineador Forzado de Montreal (Montreal Forced Aligner) / Agregación de Características Multicapa (Multilayer Features Aggregation)
CWT Transformada de Ondícula Continua (Continuous Wavelet Transform)
HMM Modelo Oculto de Markov (Hidden Markov Model)
RNN Red Neuronal Recurrente (Recurrent Neural Network)
NMT Traducción Automática Neuronal (Neural Machine Translation)
LO Locutor Objetivo
LOs Locutores Objetivos
LI Locutor Impostor
LIs Locutores Impostores
LOS Locutor Objetivo Sintetizado
LOSSs Locutores Objetivos Sintetizados
UBM Modelo de Fondo Universal (Universal Background Model)
JFA Análisis Factorial (Joint Factor Analysis)
DNN Red Neuronal Profunda (Deep Neural Network)
GAN Red Adversaria Generativa (Generative Adversarial Network)
CNN Red Neuronal Convolucional (Convolutional Neural Network)
LSTM Memoria a Largo y Corto Plazo (Long Short-Term Memory)
fdp Función de Densidad de Probabilidad (Probability Density Function)
ML Aprendizaje Automático (Machine Learning)
GML Aprendizaje Automático Generativo (Generative Machine Learning)

Capítulo 1

Introducción, motivación, objetivos y estructura del documento

1.1. Introducción y motivación

En la actualidad, la tecnología desempeña un papel fundamental en todos los aspectos de la sociedad, impulsando avances significativos que conllevan un crecimiento a niveles sin precedentes.

La Inteligencia Artificial (IA), se destaca como uno de los avances más relevantes en la actualidad, ya que desafía los paradigmas de la informática tradicional. Mientras que antes los ordenadores necesitaban instrucciones detalladas para realizar tareas específicas, la IA permite que aprendan a resolver problemas por sí mismos, mediante un proceso de aprendizaje continuo.

El proceso de aprendizaje, central en la IA, implica que los sistemas aprendan de la experiencia y mejoren su rendimiento con el tiempo. Por ejemplo, en sistemas de identificación de imágenes, el ordenador no necesita ser programado para reconocer objetos, sino que aprende a hacerlo a partir de una amplia variedad de ejemplos.

Debido a la fuerza disruptiva que implica la IA, han aparecido múltiples herramientas asociadas a esta tecnología, como por ejemplo, generación de código, clonación de voz, asistentes virtuales, reconocimiento facial, traducción automática, diagnóstico médico, conducción autónoma, etc. Es posible percatarse debido a todas las funcionalidades que tiene la inteligencia artificial, que se ha convertido en una tecnología indispensable en la sociedad actual.

La voz resulta útil para ciertas tareas, como para identificar locutores, reconocimiento del habla, detección de voz, síntesis de texto a voz, ‘diarization’ (identificación de los diferentes locutores que componen un diálogo), etc.

El reconocimiento del locutor puede servir para autenticar el acceso a ciertas aplicaciones. Por ejemplo, el banco HSBC utiliza la autenticación por voz para que los clientes accedan a sus cuentas bancarias a través de la aplicación móvil, el servicio telefónico, el análisis de voz forense, etc. Existen multitud de métodos para realizar este objetivo, donde los más destacados actualmente son los métodos basados en aprendizaje profundo.

Han aparecido avances en la investigación de arquitecturas de redes neuronales utilizadas en el reconocimiento de locutores, como las redes ECAPA-TDNN. Esta

arquitectura presenta un mejor rendimiento en comparación con arquitecturas anteriores. En cuanto a los avances en funciones de pérdida, cabe destacar la pérdida generalizada de extremo a extremo (GE2E), la cual ha demostrado mejorar la capacidad de los modelos para distinguir entre múltiples locutores. Las funciones de pérdida determinan qué tan bien el modelo está aprendiendo durante el proceso de entrenamiento.

Respecto a la síntesis de texto a voz, se trata de generar unas ondas que sintetizan el habla en cuestión, a partir de un texto como entrada. Donde se quiere una caracterización de dicha voz, si se plantea que sea una voz más grave o más aguda, más alegre, o más formal, etc. Un ejemplo funcional muy importante de esta técnica es que, a partir de un texto en un lenguaje determinado, se generen múltiples voces, en diferentes idiomas, con voces personalizables, para así realizar el doblaje de una película, por ejemplo. Por tanto, esta tecnología presenta un gran beneficio en diferentes industrias, como en el cine, videojuegos, política. El uso de la IA para la síntesis de voz en diferentes idiomas puede aportar grandes beneficios en la traducción en tiempo real durante sesiones plenarias. Una IA bien entrenada tiene la capacidad de abstraer sentimientos, entonaciones y otros matices de la voz humana, asegurando que la traducción sea precisa y profesional, evitando malinterpretaciones de lo que los diferentes ponentes desean comunicar.

Durante las comisiones y plenos, la forma en que se dice algo, la entonación y los sentimientos expresados son cruciales para entender el mensaje completo. La IA puede analizar y replicar estas características proporcionando una traducción fiel que refleja no solo las palabras, sino también la intención detrás de ellas. Esto es esencial para mantener la integridad del mensaje y asegurar que todos los participantes comprendan correctamente lo que se está diciendo.

Sin embargo, existe un lado negativo en todas estas tecnologías, y es el caso de la generación de voz clonada, porque puede llegar a suplantar personas y generar tácticas de ingeniería social con fines ilegales. Las tecnologías utilizadas actualmente para generar modelos de creación de voz sintetizada, basadas en aprendizaje profundo, son modelos de extremo a extremo como Tacotron, estos modelos implican una mejora significativa que con respecto a otro tipo de modelos. Arquitecturas basadas en Transformers llevan una mejor controlabilidad y eficiencia de entrenamiento, esto se podrá ver en la arquitectura utilizada para realizar el sistema de TTS, el cual se denomina FastSpeech 2.

Cuando se quiere modelar la voz, es necesario que la calidad de audio sea buena, para así generar modelos adecuados, aunque, por otra parte, si se quisiera entrenar un modelo de reconocimiento de locutor, sería adecuado entrenar el modelo con ruido asociado a la voz, para así entrenar más fielmente el modelo, y poder reconocer al locutor en entornos más diversos (ruidosos, con diferente acústica, etc). La calidad de la base de datos del locutor debe ser muy buena, para así poder abstraer todas las variabilidades que puedan caracterizar la voz a modelar. Debido a esta variabilidad, para realizar la modelación con una minimización de cantidad de datos a entrenar, sin perjudicar gravemente a la calidad de la modelación, se puede plasmar una base de datos tal que maximice la diversidad de fonemas. Es posible mejorar la funcionalidad de los sistemas de transcripción de texto a voz utilizando sistemas de transcripción de texto a voz basado en múltiples voces (Multi-Speaker TTS) (se estudiará dicho sistema, basado en FastSpeech 2, en la fase experimental del proyecto).

1.2. Objetivos

El objetivo de este trabajo es evaluar la viabilidad de los sistemas para conversión ‘Voice Cloning’ y síntesis automática de voz (TTS- Text To Speech) actuales y de código abierto para la generación de señal fraudulenta de voz. Esta voz falsificada se emplea para “engañosar” un sistema biométrico, utilizando para ello técnicas de ‘spoofing’. Este hecho supone un peligro cada vez mayor debido a la creciente disponibilidad de sistemas de síntesis de voz y conversión de voz (clonación de voz) de muy alta calidad. En este trabajo se obtendrá voz falsificada a partir de muestras de voz donantes disponibles en internet. Para ello se utilizarán las tecnologías más avanzadas de código abierto disponibles en el estado del arte.

Se desarrollarán dos sistemas de clonación de voz capaces de sintetizar voz de manera realista: el primer sistema está compuesto por **Fastspeech 2** [45] y **Diff-Wave** [25], y el segundo sistema por **Tacotron 2** [51] y **HiFi-GAN** [24]. Además, se van a estudiar tres sistemas diferentes de verificación de locutor: **x-vector** [56], **r-vector** [16] y **ECAPA-TDNN** [5], para determinar la identidad del locutor y analizar la robustez y rendimiento de dichos sistemas ante técnicas de ‘spoofing’, como suplantación de identidad del locutor inscrito en el sistema ASV.

1.3. Estructura del documento

En esta sección se describen los diferentes capítulos que componen el trabajo.

- Capítulo 1: Introducción, motivación, objetivos y estructura del documento. En este capítulo, se analiza cómo la IA está transformando el campo del reconocimiento y síntesis de voz, sus aplicaciones actuales, avances recientes, y también los desafíos que enfrenta. Además, se presentan los diferentes objetivos del trabajo y una descripción de la estructura del documento.

- Capítulo 2: Reconocimiento del Locutor y Síntesis de Voz: Técnicas y Modelos Avanzados.

En este capítulo, se analiza teóricamente el reconocimiento del locutor profundizando en la verificación del locutor, además, se explican los diferentes extractores de características del locutor y las medidas utilizadas para establecer el rendimiento y eficiencia de los sistemas de ASV. Por otra parte, se analizan los sistemas de TTS, para ello se estudian diferentes modelos basados en transcripción de texto a espectrograma Mel y diferentes vocoders, modelos basados en la transcripción del espectrograma Mel a formas de onda.

- Capítulo 3: Experimentos.

En este capítulo, se presentan los resultados de los diferentes experimentos realizados. Se estudia la robustez de los sistemas ASV frente a locutores que intentan realizar ataques de suplantación de identidad, comparándolos con aquellos que no lo hacen, utilizando clonaciones de voz correspondientes para las pruebas.

- Capítulo 4: Conclusiones.

En este capítulo, se desarrollan las diferentes conclusiones a las que se llega a raíz de los resultados obtenidos en los experimentos. Conclusiones enfocadas en analizar la robustez de los modelos estudiados.

Capítulo 2

Fundamento Teórico

2.1. Introducción

En este capítulo se realizará el estudio teórico de los sistemas de TTS y ASV utilizados en la fase experimental de este trabajo. Se profundizará en las técnicas y arquitecturas empleadas en los diferentes sistemas.

Por una parte, se explica qué es el reconocimiento de locutor, enfocándose en la verificación del locutor automático. Para ello, se describen los dos tipos de sistemas existentes, las fases de funcionamiento del sistema, el spoofing en este ámbito, los sistemas de extracción de características del locutor utilizados experimentalmente, que son x-vector, ResNet y ECAPA-TDNN y las medidas de rendimiento y presentación de resultados, centrándose en la tasa de error equivalente (EER), que es una medida de precisión y las curvas ROC (Característica Operativa del Receptor) y DET (Compensación de Errores de Detección).

Por otra parte, se describe qué es la síntesis de voz, profundizando en los diferentes sistemas generadores de habla sintética. En primer lugar, se explica Tacotron 2, un modelo de TTS extremo a extremo capaz de generar la forma de onda del habla sintetizada directamente a partir del texto de entrada. A continuación, se presenta FastSpeech 2, un modelo basado en Transformers que genera un espectrograma Mel a partir del texto de entrada.

Además, se explican dos vocoders, que son los encargados de generar la forma de onda a partir del espectrograma Mel: HiFi-GAN y DiffWave.

2.2. Reconocimiento del locutor

Según [29], el reconocimiento automático del locutor es la tarea de identificar o verificar la identidad de un individuo a partir de muestras de su voz utilizando algoritmos de aprendizaje automático, sin ninguna intervención humana. El objetivo es comprender quién está hablando a partir de sus discursos. El habla natural no es simplemente una secuencia de sonidos, sino una combinación de diferentes sonidos que a menudo se entrelazan sin límites distintos. Esto significa que las palabras y los sonidos no están claramente delimitados, sino que se solapan y se mezclan de

manera continua.

Para abordar el problema del reconocimiento del locutor, se han desarrollado varias tecnologías y técnicas. En [12], se describen algoritmos para extraer características útiles para esta tarea. Este trabajo, se enfoca en el uso de sistemas de IA, específicamente en el aprendizaje automático y el aprendizaje profundo.

En la Figura 2.2 se muestran los dos modos de funcionamiento de un sistema de reconocimiento automático de locutor, basados en [47]. En la Figura inferior, se ilustra el esquema general de verificación de locutor, donde en la entrada se presenta un vector de características, denominado *embedding*, el cual representa al locutor fuente. En dicho sistema, se tiene un registro de los locutores inscritos, representados mediante *embeddings*. El *embedding* del locutor fuente se compara con el *embedding* del locutor inscrito. Si el resultado es mayor o igual que cierto umbral, se acepta al locutor fuente y si el resultado es menor, se rechaza al locutor fuente.

En la Figura superior, se ilustra el esquema general de identificación del locutor, en la entrada del sistema se presenta un *embedding*, este se compara con los múltiples *embeddings* de los locutores de referencia existentes en el identificador, y se selecciona el locutor inscrito que máxima probabilidad presente (siempre y cuando supere cierto umbral de probabilidad), generando a la salida la identificación del locutor de entrada.

En este proyecto únicamente se analizará experimentalmente la verificación automática del locutor (ASV).

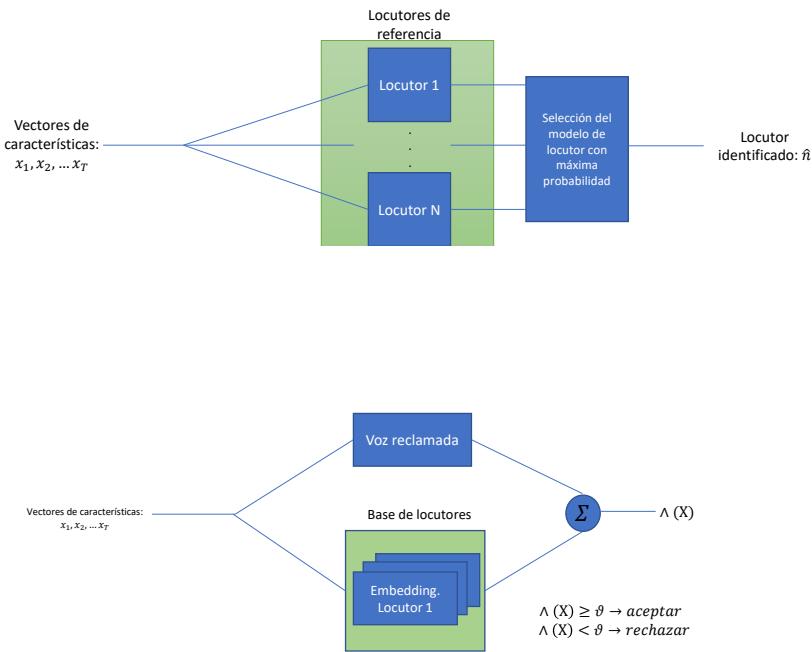


Figura 2.2: Esquema de identificación y verificación del locutor. Imagen adaptada de [47]

2.2.1. Identificación del locutor

Según [61], la identificación del locutor es el proceso de identificar al locutor a partir de un enunciado determinado, comparando la biometría de voz del enunciado con los modelos de enunciado almacenados de antemano. Básicamente, se está clasificando una voz particular para identificar a una persona. Esto se puede hacer, por ejemplo, utilizando modelos de aprendizaje profundo, aunque no existe un método matemático específico para clasificar a una persona en una conversación. Como ejemplo práctico, existe en Zoom o Google Meet una sección dentro del cuadro personal que se resalta cuando una persona en concreto habla. Esto se hace mediante la identificación del locutor aplicada en estas plataformas.

Una de las aplicaciones principales de la identificación del locutor es la diarización. Según [1], la diarización de locutores es el proceso de etiquetar a diferentes locutores en una secuencia de audio, respondiendo a la pregunta “quién habló cuándo” en una conversación de varios locutores. Se parte de una conversación con múltiples locutores y se realiza una segmentación de dicha conversación, en función del locutor que esté hablando y se identifican los segmentos que corresponden a cada locutor. Aunque se desconozca la identidad real de los locutores, a la salida de dicho sistema se obtiene un etiquetado de la señal de audio de entrada, en función de los diferentes locutores sin importar su identidad.

La identificación de locutor se basa en una comparación de “uno a muchos” locutores. Se tiene como entrada un segmento de voz desconocido, y es necesario saber quién está hablando. Dicho segmento de voz se debe comparar con los N modelos existentes, referentes a diferentes locutores. Para cada modelo disponible, se obtiene una determinada puntuación la cuál será más alta cuanto mayor sea la probabilidad de que el segmento de voz coincida con el modelo de locutor comparado. La salida del identificador será el modelo que plasme mayor probabilidad de coincidencia, es decir, la máxima puntuación.

2.2.2. Verificación del locutor

En este apartado se explica brevemente qué es la verificación del locutor, las técnicas utilizadas para extraer las características acústicas del locutor y las dos modalidades principales de sistemas de verificación del locutor. Después, se explican las fases del proceso de verificación del locutor y finalmente, los diferentes tipos de *spoofing* que existen y un poco de historia del origen y actualidad de la verificación del locutor contra *spoofing*.

Según [57], la verificación del locutor es la tarea de autenticar la identidad declarada de un locutor, basándose en alguna señal de voz y en el registro del locutor inscrito. Es decir, es una técnica mediante la cual se puede verificar que un locutor es quien dice ser. Se basa en extraer las características acústicas del locutor, a nivel de enunciado y a nivel de puntuación.

Para extraer las características a nivel de enunciado básicamente lo que se hace es transformar la forma de onda del locutor en una representación paramétrica, también denominado *embedding*, el cual describe las características prosódicas de la

voz del locutor. Para ello se suelen utilizar, por ejemplo, los coeficientes cepstrales de frecuencia Mel (MFCC) y modelos de mezcla Gaussiana (GMMs).

La caracterización a nivel de puntuación se basa en comparar el *embedding* generado con el *embedding* inscrito en el sistema. Si la comparación cumple un cierto umbral de puntuación, se establece la aceptación del locutor fuente en el sistema de verificación.

Una forma de verificar que el locutor es quién dice ser es, por ejemplo, analizar la puntuación asociada a la comparación con su modelo asociado, analizar la puntuación con un modelo incorrecto, y la diferencia de ambas puntuaciones, si superan un cierto umbral se considera que el locutor es aceptado en el sistema de verificación. Si no se supera dicho umbral, se rechaza al locutor.

Un sistema de verificación de locutor puede ser dependiente del texto introducido o puede no serlo. En un sistema de texto dependiente, es preciso pronunciar un texto en concreto, para que se pueda aceptar al locutor en cuestión, mientras que, en un sistema de verificación de locutor con independencia del texto, puede aceptar un locutor con cualquier frase dicha. Puede decirse que el verificador de locutor se puede diferenciar en función de las restricciones que se imponen al texto del habla utilizado. Según [7], las técnicas de reconocimiento de locutores se pueden dividir en dos modalidades principales:

- **Sistema dependiente del texto:** En este sistema, el locutor tiene que decir un texto concreto, por tanto, el sistema queda restringido a funcionar ante un texto en particular. En una aplicación de autenticación, por ejemplo, se va a precisar que se diga una combinación de palabras en específico o una ya preestablecida para que el sistema verifique la voz, por tanto, el sistema ya sabe lo que se va a decir, por consiguiente, esta tarea es más sencilla que en un sistema independiente del texto.
- **Sistema independiente del texto:** El sistema independiente del texto es más flexible, por consiguiente, es más útil que el sistema dependiente del texto. En este caso, no existe restricción con respecto a lo que se dice. Por tanto, este sistema es más funcional en lo que respecta al reconocimiento del locutor, puesto que es posible identificarlo, independientemente de lo que diga. Para llevar a cabo dicho reconocimiento, es necesario crear una “huella” de dicho locutor, la cual debe ser totalmente diferenciable de los demás locutores que puedan estar modelados en dicho sistema. Para ello, se genera una representación numérica de la voz del locutor conocida como ‘embedding’, donde cada locutor de estudio tendría un *embedding* asociado. Gracias a esta huella, es posible identificar a un locutor, independientemente de lo que diga. Para poder generar el *embedding* es algo complejo, puesto que debe ser único, aunque dos voces sean muy semejantes entre sí.

En este proyecto se estudia la verificación de locutor basada en independencia del texto introducido.

2.2.2.1. Fases del proceso de verificación del locutor

En este apartado se explican las tres fases que componen el proceso de verificación del locutor, según [63]. Primeramente, se realiza la fase de desarrollo, en la cual se entrena el modelo para definir la población de locutores. Una vez entrenado el modelo, es posible instanciar la inscripción de un nuevo locutor, esto se realiza en

la fase de inscripción. Finalmente, para verificar la identidad del locutor inscrito se realiza la fase de evaluación.

- **Fase de desarrollo:** En la fase de desarrollo, el modelo de fondo se entrena a partir de una gran recopilación de datos para así definir la variedad de locutores. Es esencial para capturar las características generales de los locutores que no están inscritos en el sistema. Puede tratarse de modelos de fondo universal, UBM [46], los cuales utilizan una combinación de distribuciones Gaussianas para representar la variabilidad de las características del habla en una población grande. Estos modelos no se centran en un locutor en específico. Existen modelos más complejos basados en análisis factorial (JFA) [22], los cuales consideran la variabilidad entre locutores y la variabilidad del locutor.
- **Fase de inscripción:** En la fase de inscripción, se registran nuevos locutores derivando información específica del locutor para obtener modelos dependientes del locutor. Se ajustan los parámetros del modelo de fondo utilizando las características del habla del nuevo locutor para así crear un nuevo modelo. Es decir, se crea un modelo específico para cada nuevo locutor.
- **Fase de evaluación:** En la fase de evaluación, se evalúa la muestra de prueba utilizando los modelos de locutores inscritos y los modelos de fondo. Se trata de determinar si el locutor de muestra pertenece a un locutor inscrito. Para ello, se establece una puntuación de similitud asociada al modelo de fondo y otra puntuación asociada al modelo de locutor inscrito. En el caso de que la puntuación de similitud dada en el modelo de locutor inscrito sea significativamente mayor que en el modelo de fondo, se acepta la identidad reclamada.

2.2.2.2. Verificación del locutor contra ataques de suplantación de identidad (spoofing)

Los sistemas de verificación de locutor son robustos cuando un impostor no hace ningún esfuerzo por parecerse al locutor real. Aunque, cuando sí existe cierto esfuerzo, el sistema verificador puede ser engañado [2].

Los ataques de falsificación de voz se clasifican, según [71], de la siguiente manera:

- **Impersonación:** Es uno de los métodos más obvios de suplantación de identidad y se refiere a ataques que utilizan voces alteradas por humanos, también conocidas como imitación humana. [15].
- **Reproducción:** Se utiliza el discurso pregrabado del hablante objetivo [65].
- **Síntesis de texto a voz (TTS):** El sistema computacional sintetiza la voz del locutor objetivo. Existen múltiples métodos de TTS actuales que generan un habla de alta calidad con un alto contenido prosódico, basado en redes neuronales profundas (DNNs) y redes adversarias generativas (GANs), por ejemplo, Tacotron [69].
- **Conversión de voz (VC):** Se utiliza la voz de una persona para generar la voz del locutor objetivo [23].

Inicialmente, en la investigación anti falsificación no se tenía un estándar de investigación hasta que se organizó un desafío de verificación de locutores suplantados

(ASVSpoof) en el congreso Interspeech en su edición de 2015 [72], donde se comenzó a utilizar un esquema de trabajo unificado, con determinados protocolos y métricas comunes, puesto que es lo que se decidió en la edición del año 2013.

La base de datos proporcionada en el desafío contiene habla genuina y suplantada. El habla genuina consiste en el habla de 106 locutores humanos (45 hombres y 61 mujeres). El habla suplantada es la forma modificada del habla genuina utilizando los métodos TTS y VC.

En 2016, en el desafío ‘Biometrics: Theory, Applications, and Systems’ (BTAS) se proporcionó una base de datos con ataques de falsificación de reproducción, TTS y VC. Esto implicó un crecimiento y visibilidad en la investigación de este área.

Después, en 2017, debido a la existencia de grabadoras de alta calidad, la segunda edición del desafío de verificación de locutores suplantados (ASVSpoof) en el congreso Interspeech, se centró en los ataques de reproducción.

Por último, mencionar que en una sesión en el congreso Interspeech en su edición 2023 [49], se presenta el contexto de cómo los conformers (modelos avanzados que combinan elementos de las redes neuronales convolucionales (CNN) y los mecanismos de atención de los Transformers), con su capacidad avanzada de modelado, pueden ser aplicados a la detección de suplantación en verificación de locutores, y cómo la utilización de técnicas de aprendizaje auto-supervisado y modelos pre-entrenados puede superar las limitaciones impuestas por la disponibilidad de datos etiquetados.

2.2.3. Extractores de características del locutor

En este apartado se detallará qué es el *embedding* del locutor, las dos categorías principales de *embeddings* y se introducirán brevemente los *embeddings* más comunes que se han utilizado para el reconocimiento de locutores. En las siguientes subsecciones se profundizará en los extractores de características del locutor utilizados en la fase experimental de este trabajo, que son los sistemas x-vector, ResNet y ECAPA-TDNN.

Según [30], se define el *embedding* del locutor como la representación de dimensión fija del enunciado de longitud variable. Los *embeddings* de los mismos locutores están cerca unos de otros en el espacio vectorial del *embedding*, y por tanto, permite una fácil comparación entre los locutores utilizando operaciones geométricas simples, por ejemplo, la distancia coseno.

En función de cómo se realice el entrenamiento, se pueden dividir los *embeddings* del locutor en dos categorías, *embedding* no supervisado, como es el caso de i-vector y el supervector del modelo de mezcla Gaussiana (GMM supervector), y *embedding* supervisado, como x-vector. La categoría de *embedding* no supervisado utiliza un modelo generativo entrenado con un criterio no supervisado de máxima verosimilitud, mientras que la categoría de *embedding* supervisado se basa en el etiquetado de los datos para obtener así un entrenamiento discriminante, normalmente se utiliza una pérdida discriminante de locutores “multi-class”.

Cuando se utiliza un sistema de verificación de locutor con independencia del texto hablado introducido, es necesario para poder verificar la identificación del locutor, generar un *embedding* asociado. Como se ha visto anteriormente, el *embedding* se trata de una representación numérica única, asociada a la voz de una persona en

concreto.

Los *embeddings*, como los i-vectors, d-vectors y x-vectors, han demostrado su superioridad en el reconocimiento del locutor [63]. Los investigadores han encontrado que los *embeddings* del locutor pueden resolver el problema de la longitud variable de las características a nivel de trama, además de codificar la identidad del locutor y el contenido del discurso en gran medida [68]. Los *embeddings* extraídos se basan en el modelo pre-entrenado de reconocimiento del locutor.

Los i-vectors son representaciones compactas de baja dimensión que se utilizan para tareas de reconocimiento como verificación e identificación del locutor. Inicialmente, surgieron para la verificación de locutores [50]. Posteriormente, el marco de i-vector se aplicó en el reconocimiento de locutores [20]. Una breve descripción del algoritmo de extracción de i-vector es la siguiente: Primeramente, se entrena un UBM y luego se obtienen modelos específicos del locutor fuente, adaptando el UBM mediante un enfoque de adaptación máximo a posteriori (MAP). De esa manera, el *embedding* i-vector considera las características del canal de grabación.

Los d-vectors son modelos convencionales de redes neuronales profundas (DNNs) que funcionan como extractores de características a nivel de trama. A la entrada de la DNN se presenta el audio, y tras ser procesado por el modelo, los resultados se promedian en la última capa oculta de la red, obteniendo así el d-vector.

Se demuestra que los d-vectors superan a los i-vectors en verificación de locutor independiente del texto en segmentos de habla cortos [55].

Posteriormente, surgen los x-vectors, los cuales se han aplicado en el reconocimiento de locutor [54]. Los x-vectors se obtienen a partir de una red neuronal de retardo de tiempo (TDNN) con submuestreo para así tener una red codificadora. Las redes codificadoras de x-vectors incluyen las siguientes categorías: TDNN [67], TDNN extendida (E-TDNN) [54], TDNN factorizada (F-TDNN), con conexiones de salto [37] y Resnet 2D [16]. Los experimentos muestran que los x-vectors pueden capturar el contenido hablado y la información relacionada con el canal [42].

2.2.3.1. x-vector

El sistema x-vector [55], se trata de un sistema de reconocimiento del locutor basado en una DNN Feed-Forward que mapea segmentos de voz de longitud variable a *embeddings*, el *embedding* se denomina x-vector y se clasifica mediante un clasificador Gaussiano pre-entrenado [33].

En la Figura 2.3 se ilustra la arquitectura del sistema x-vector. Inicialmente, a la entrada se tiene el audio segmentado en un número determinado de tramas, denotadas como T . Se define cada trama como x_i , con $i = 1, 2, \dots, T$. El audio entramado se proyecta inicialmente en una TDNN, esta red es similar a cualquier CNN o red neuronal recurrente (RNN) de aprendizaje profundo. En esta red se capturan las características a nivel de trama del audio de entrada. A la salida de la TDNN, se generan T vectores de características a nivel de trama, cada uno de dimensión N . Luego, utilizando una capa de agrupación de estadísticas, se calcula la media (μ) y la desviación estándar (σ) de cada dimensión de la secuencia de vectores de características. La salida son $2N$ estadísticas, las cuales se proyectan en un bloque de capas ocultas que extraen características a nivel de enunciado. El vector final obtenido es el *embedding* x-vector.

Finalmente, se presenta una capa con una función exponencial normalizada (o

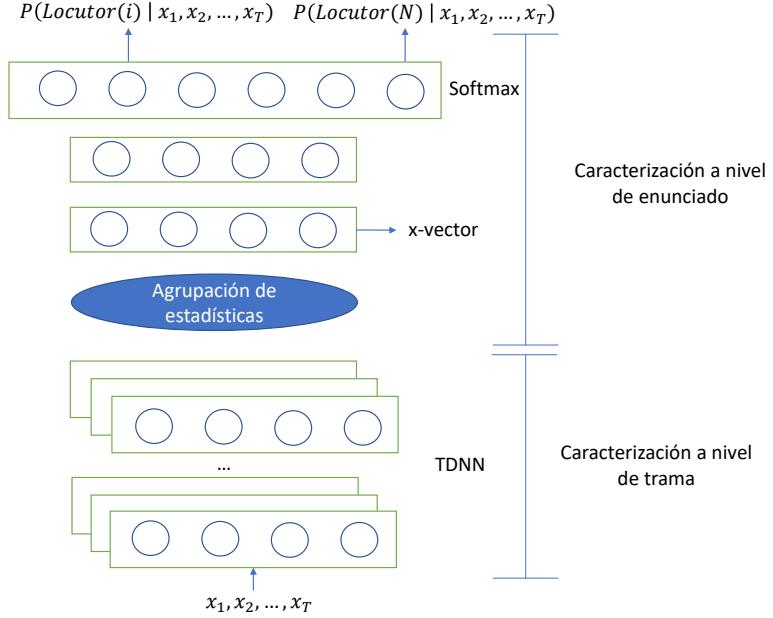


Figura 2.3: Arquitectura del sistema x-vector. Imagen adaptada de [53]

función Softmax), la cual toma el x-vector y predice la probabilidad de cada locutor dada la sentencia, denotada como $P(\text{locutor}(i)|x_1, x_2, \dots, x_T)$.

A continuación, se va a explicar el funcionamiento de TDNN, para así comprender cómo se realiza el procedimiento de codificación en las capas a nivel de trama.

TDNN

La red neuronal de retardo de tiempo (TDNN) convierte una señal acústica en una representación fonética. Se trata de una red neuronal convolucional (CNN) unidireccional (1D) sin agrupación y con dilataciones. La formalización matemática de CNN general se describe en [21] y [6]. Es utilizada en reconocimiento de voz como “Kaldi” [38], donde se convierte una señal de voz en una secuencia de fonos.

El procedimiento que se lleva a cabo en la TDNN es el siguiente: Se analiza el audio segmentado en tramas, donde cada trama tiene la duración típica de un fonema, y se predice el fonema más probable de cada trama. Esto es posible porque en función del idioma utilizado existen unos fonemas predefinidos.

La entrada a la TDNN es una matriz, donde el vector columna representa un determinado segmento de audio, denominado $x_t \in R^m$, siendo m la dimensión del segmento de audio y las filas son los valores de características acústicas asociados al segmento de audio (coeficientes cepstrales de frecuencia de Mel (MFCCs)). Esta matriz se denomina “matriz de características a la entrada”, $X \in R^{m \times t}$, donde t es el número de segmentos de audio.

La red utiliza una matriz de pesos sintonizables de tamaño inferior a X denominada Kernel o filtro, $W \in R^{m \times l}$, l es el número de columnas del Kernel, siendo $l < t$. El Kernel se desliza sobre la matriz de entrada y la transforma en una salida realizando la operación de convolución. La región que cubre W dentro de la matriz de características a la entrada se denomina “Campo receptivo”. Si se supone que se tiene un “padding” nulo, $p = 0$ y el Kernel tiene un deslizamiento de una posición a

la vez $s = 1$. Por consecuencia, el ancho del vector a la salida (o) será:

$$o = \left(\frac{t - l + 2p}{s} \right) + 1 \quad (2.1)$$

En cada instante de tiempo t se realiza una convolución, se agrega un sesgo entrenable b , y se aplica una función de activación no lineal φ para obtener el vector de salida. En un determinado paso q del Kernel, cuando este es continuo (no dilatado), la salida y_q queda de la siguiente manera:

$$y_q = \varphi([W] * [X_q] + b) \quad (2.2)$$

Donde $*$ es la operación de convolución y X_q es la matriz de entrada en el campo receptivo.

Cuando el Kernel no se aplica a vectores de características contiguos, se trata de un análisis entramado tipo “contexto”, denominado proceso “dilatado” o “submuestreado”. En esta situación, el ancho a la salida o , se ve variado con respecto a la situación de Kernel continuo.

$$o = \left(\frac{t - (pos_j^{max} - pos_j^{min}) + 2p}{s} \right) \quad (2.3)$$

La salida establecida en la Ecuación 2.2 sirve para cuando el Kernel es discontinuo, simplemente hay que ajustar la matriz de entrada en cada paso $[X]_q$, para que se cumpla el campo receptivo dado.

Anteriormente, a la salida tras realizar los o pasos, se tenía un vector $y \in R^{1 \times o}$. Si se tiene un número determinado de filtros N , a la salida se obtendrá una matriz $Y \in R^{N \times o}$. Si esta matriz resultante se aplica a una nueva TDNN, y así sucesivamente se tiene el fundamento en el que se basa la codificación en las capas a nivel de trama del sistema x-vector.

Surgen modificaciones en la arquitectura TDNN puesto que plantea un rendimiento muy bueno a la hora de crear sistemas de identificación de locutor. Entre ellos, se encuentra Resnet y ECAPA-TDNN. En este trabajo se utilizan como redes codificadoras TDNN, Resnet y ECAPA-TDNN. En las siguientes subsecciones se explicarán las redes codificadoras Resnet y ECAPA-TDNN.

2.2.3.2. ResNet

En ResNet, la red codificadora TDNN de x-vector se reemplaza por la red residual de 34 capas (Resnet34), descrito en [16], además, las características entrantes al sistema, es decir, los MFCCs son reemplazados por bancos de filtros log-Mel y la codificación se basa en convoluciones bidimensionales (CNN-2D), en vez de CNN-1D.

Anteriormente, se utilizaban capas densas, como “GoogleNet” [59], y VGG16 que es una arquitectura de red neuronal profunda desarrollada por Google en 2014. Estaba formada por centenares de capas. Al tener una gran cantidad de capas, esto implica un aumento en el coste computacional de la red (conforme más capas, más difícil de entrenar es la red), es por ello, que ante la necesidad de reducir de la cantidad de capas a utilizar, surge ResNet, que utiliza decenas de capas. Además, se soluciona el problema del “desvanecimiento del gradiente”, superando así a las redes anteriormente propuestas.

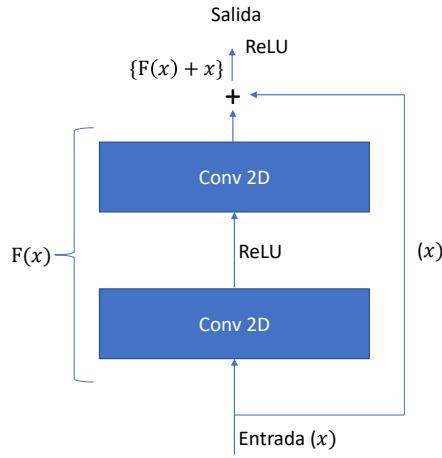


Figura 2.4: Bloque residual convolucional 2D utilizado como bloque básico en ResNet. Figura adaptada de [66].

Durante el entrenamiento de una red neuronal, se utiliza un proceso llamado retropropagación (backpropagation) para ajustar los pesos de la red con el objetivo de minimizar el error. En este proceso, se calcula el gradiente del error con respecto a cada peso, lo que indica en qué dirección y cuánto debe ajustar los pesos para reducir el error. Sin embargo, en redes neuronales profundas (DNN), donde hay muchas capas, los gradientes pueden volverse extremadamente pequeños a medida que se propagan hacia atrás desde la capa de salida hasta las capas iniciales. Como solución, se hace la normalización de los gradientes para así resolver el problema del “desvanecimiento del gradiente”, pero aun así sigue existiendo un alto coste computacional.

Se parte de una red en estado de entrenamiento, la cuál esta convergiendo y se mide la precisión, dicha precisión aumenta conforme se entrena la red. Si se agregan nuevas capas a la red, a partir de un determinado número de capas, el rendimiento de la red no mejora, sino que empeora la métrica de error en el conjunto de entrenamiento, y por tanto, este efecto no es deseable. Pero, se comprueba experimentalmente, que si se implementa una conexión de salto con una función identidad asociada, esta no afecta negativamente al desempeño de la red, sino que plantea ciertos beneficios, según [17].

Si a una red neuronal con un número determinado de capas N , se le intercalan X capas que calculan la función identidad, la red resultante con $N + X$ capas tiene la misma eficacia que la red con N capas. Para variar la cantidad efectiva de capas, se introduce el concepto de bloque residual.

El bloque residual se ilustra en la Figura 2.4. Se compone de una ruta residual compuesta por dos capas de pesos sinápticos intercalados por una función de activación de Unidad Lineal Rectificada (ReLU¹), designada como $F(x)$ y una ruta denominada conexión de salto por “identidad”, designada como x (entrada del bloque). Ambas rutas se suman y se aplica la función ReLU nuevamente para obtener la salida del bloque. En dicho bloque, la información y el gradiente en sentido inverso atraviesan ambas rutas.

Se construyeron redes residuales de diferentes tamaños y todas ellas superaban a las arquitecturas VGG16 [52] y GoogleNet. Esto se cree que se debe a dos razones

¹ReLU es una función de activación introducida por [14]. Funciona aplicando un umbral en los valores en 0, es decir, $f(x) = \max(0, x)$.

principalmente:

- El gradiente de la función de pérdida puede retroceder hacia la entrada a través de la conexión atajo, implicando en una menor degradación del gradiente.
- Se demuestra en múltiples artículos que la mezcla de diferentes tipos de información semántica, es decir, la suma de la información que pasa por las capas de convolución con la que no, presenta un aumento en el rendimiento de los sistemas de reconocimiento visual, aunque se puede extrapolar al ámbito auditivo.

En el entrenamiento de la red ResNet se suele utilizar profundidad estocástica para evitar la degradación del gradiente, esto se realiza desactivando ciertas capas, haciendo que la función de pérdida retroceda por la conexión de salto.

2.2.3.3. ECAPA-TDNN

La topología del sistema x-vector (TDNN) mejoró aplicando elementos de la arquitectura ResNet, puesto que se demuestra que agregar conexiones residuales entre las capas a nivel de trama mejora la generación de *embeddings*. Además, las conexiones residuales permiten que el algoritmo de retropropagación converja más rápido, ayudando así al problema del desvanecimiento del gradiente [17].

En el sistema x-vector , la capa de agrupación de estadísticas trata todas las salidas a nivel de trama de su capa oculta anterior por igual. Sin embargo, en el discurso, existen algunas partes qué son útiles para discriminar la identificación del locutor, como ciertos contenidos fonéticos y existen otras partes que no lo son, por ejemplo, el ruido, silencio, etc. Es por ello, que esta capa de agrupación de estadísticas se reemplaza por una capa de agrupación de estadísticas con auto-atención, propuesto en [31]. Este mecanismo se utiliza para seleccionar qué tramas del discurso son útiles y cuáles no.

En la Figura 2.5 se ilustra el sistema ECAPA-TDNN, el cual se basa en la estructura del sistema x-vector. A la entrada del sistema se tiene la matriz de características, formada por un número determinado T , de tramas de audio, $X = [x_1, x_2, \dots, x_T]$. Esta matriz se proyecta en una serie de SE-Res2Blocks unidimensionales. Los bloques SE unidimensionales (bloques de compresión-excitación unidimensional) reescalán los canales (diferentes partes de la información que la red está procesando, se presentan tres canales: C_1, C_2, C_3) de los mapas de características intermedias a nivel de trama para insertar información de contexto global. Los bloques Res2 unidimensionales mejoran el rendimiento mientras reduce la cantidad total de parámetros al utilizar convoluciones agrupadas de manera jerárquica. Además, la Agregación de Características Multicapa (MFA) [8] fusiona información complementaria antes de la capa de agrupación de estadísticas concatenando el mapa de características a nivel de trama final con los mapas de características intermedios de las capas anteriores.

A la salida de la serie de SE-Res2Blocks se obtienen T vectores de características, denominado $H = [h_1, h_2, \dots, h_T]$. Donde h_t es la representación oculta de la trama de entrada x_t capturado por la capa oculta, debajo de la capa de auto-atención.

La capa de agrupamiento de estadísticas utiliza un mecanismo de atención dependiente del canal y del contexto, que permite que la red preste atención a diferentes tramas por canal.

A la salida de la capa de atención se presenta un vector compuesto por estadís-

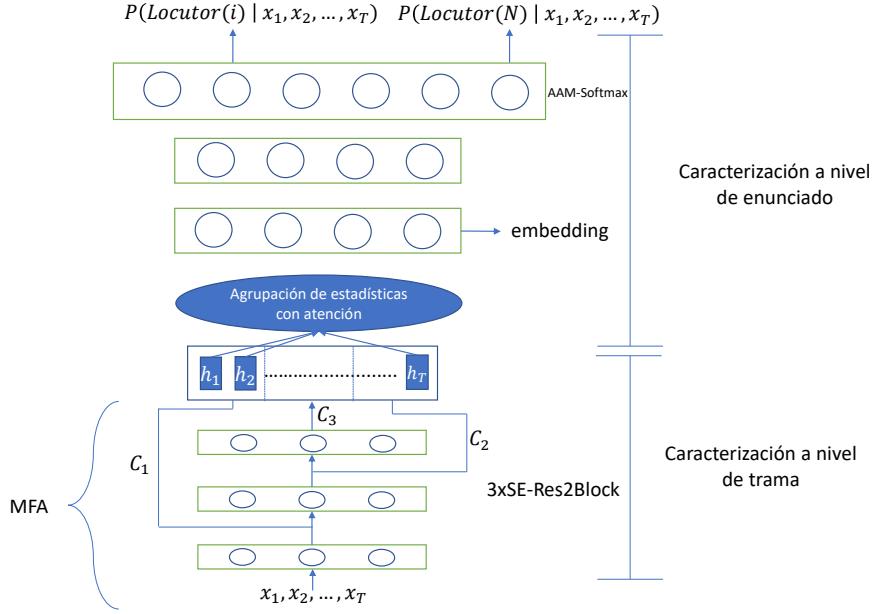


Figura 2.5: Arquitectura del sistema ECAPA-TDNN. Figura adaptada de [4].

ticas (media y desviación estándar) ponderadas.

La red se entrena optimizando la AAM-Softmax [3]. AAM hace referencia a 'margen angular aditivo'. AAM-softmax es una mejora poderosa en comparación con Softmax puesto que optimiza directamente la distancia del coseno entre los *embeddings* de los locutores.

2.2.4. Protocolo de evaluación del sistema de ASV frente a ataques de spoofing.

En esta sección se presenta un protocolo experimental genérico que se aplica a la mayoría de los trabajos de verificación de locutor automática (ASV) frente a ataques de suplantación de identidad (spoofing). Se detalla el diseño de las bases de datos y las métricas de evaluación con un enfoque en la comparabilidad de los resultados de referencia con los de estudios sobre vulnerabilidades.

2.2.4.1. Diseño del conjunto de datos

El protocolo experimental más significativo en el estudio sobre la suplantación de identidad es la comparación del rendimiento del sistema de referencia con el del mismo sistema cuando se somete a ataques de suplantación [73].

La mayoría de los estudios de suplantación se ajustan al marco general de evaluación ilustrado en la Figura 2.6. El esquema ilustra tres posibles entradas: locutor objetivo (LO), haciendo referencia a la voz genuina (usuario inscrito en el sistema ASV), locutor impostor (LI), el cual no hace ningún esfuerzo por parecerse al locutor objetivo y el locutor objetivo sintetizado (LOS), que hace referencia a la suplantación de la voz del locutor objetivo mediante técnicas de spoofing. El rendimiento de referencia del sistema de ASV se evalúa utilizando un conjunto determinado de pruebas legítimas M (a), y un conjunto determinado de pruebas impostoras N (b)

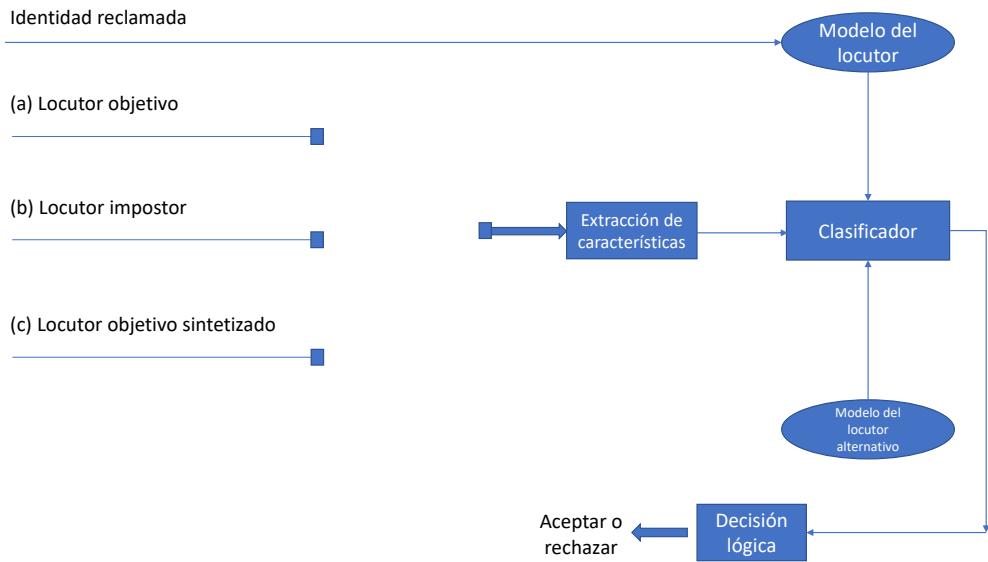


Figura 2.6: Ilustración del marco general utilizado en estudios de ataques de suplantación de identidad. Figura adaptada de [73]

, mientras que el rendimiento bajo suplantación se evalúa con un conjunto de M pruebas legítimas (a) y N pruebas de suplantación (c).

El rendimiento de referencia y el rendimiento bajo suplantación son directamente comparables, puesto que el locutor objetivo y el sistema de ASV es el mismo en ambas evaluaciones, por consiguiente, la diferencia entre ellos refleja la vulnerabilidad del sistema al ataque de suplantación considerado.

El algoritmo que se lleva a cabo para aceptar o rechazar la identidad reclamada del locutor mediante el sistema de ASV es el siguiente: se presentan dos entradas al sistema de ASV: la identidad reclamada y la voz de un locutor de prueba. A partir de la identidad reclamada, que está inscrita en el sistema, se obtiene el *embedding prototípico*, el cual se interpreta como el modelo del locutor. A partir de la voz del locutor de prueba, se utiliza un extractor de características para obtener el *embedding* correspondiente, que se interpreta como el modelo del locutor alternativo. Ambos modelos se introducen en el clasificador para generar una decisión lógica, es decir, aceptar o rechazar al locutor de prueba.

2.2.4.2. Medidas de rendimiento y presentación de resultados

El enfoque de este trabajo se centra en la autenticación mediante biometría de voz, enfocándose en el modo de verificación. Para evaluar la idoneidad de un método de autenticación biométrica para aplicaciones del mundo real, se deben utilizar tres criterios principales para evaluar el sistema, según [60], son la precisión de la verificación, la eficiencia del sistema y la facilidad de uso del sistema.

Precisión de la verificación

Las métricas utilizadas habitualmente para evaluar la precisión de la verificación de un método de autentificación biométrica, son la tasa de falso rechazo (FRR), la tasa de falsa aceptación (FAR) y la tasa de error equivalente (EER). La relación

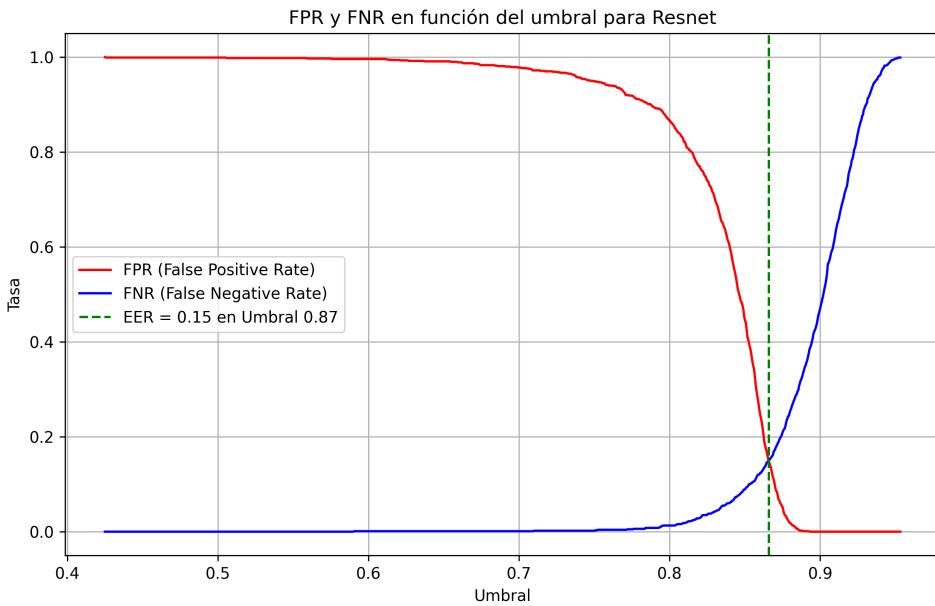


Figura 2.7: Relación existente entre FRR, FAR y EER

existente entre FRR, FAR y EER se muestra en la Figura 2.7. Se observa cómo la FAR y FRR varía en función del umbral de puntuación establecido. El punto de cruce entre las tasas de falsa aceptación y tasa de falso rechazo determina la tasa de error equivalente, establecido en un umbral de decisión específico. A continuación se definen la FRR, FAR y EER.

Para evaluar el desempeño de un problema de clasificación sobre la base de datos de prueba, se puede utilizar una matriz de confusión (Tabla 2.1).

Entrada/Salida	Valores predichos	
Valores reales	Si	No
Si	TP	FN
No	FP	TN

Tabla 2.1: Matriz de confusión

Donde, *TP* (verdadero positivo), significa que un locutor que está registrado (legítimo) en el sistema ASV, intenta acceder y el sistema lo reconoce correctamente, aceptándolo como usuario autorizado. *FP* (falso positivo), significa que un locutor que no está registrado (ilegítimo) en el sistema ASV, intenta acceder y el sistema lo reconoce incorrectamente, aceptándolo como usuario autorizado. *FN* (falso negativo), significa que un locutor que está registrado en el sistema ASV, intenta acceder y el sistema lo reconoce incorrectamente. No lo acepta como usuario autorizado. *TN* (verdadero negativo), significa que un locutor que no está registrado en el sistema ASV, intenta acceder y el sistema lo reconoce correctamente. No lo acepta como usuario autorizado.

- **Tasa de rechazo falso (FRR):** La FRR es la relación porcentual del número

de locutores legítimos que son rechazados falsamente, con respecto al número total de intentos de acceso por parte de los locutores legítimos.

$$FRR = \frac{FN}{FN + TP} \quad (2.4)$$

Donde, FN es número de falsos negativos y TP el número de verdaderos positivos. Un valor de FRR más bajo indica que hay menos usuarios legítimos siendo rechazados incorrectamente. Esto significa que el nivel de usabilidad del sistema es más alto.

- **Tasa de aceptación falsa (FAR):** La FAR es la relación porcentual del número de locutores ilegítimos que son aceptados falsamente, con respecto al número total de intentos de acceso por parte de los locutores ilegítimos.

$$FAR = \frac{FP}{FP + TN} \quad (2.5)$$

Donde, FP es número de falsos positivos y TN el número de verdaderos negativos. Un valor de FAR más bajo indica que hay menos usuarios ilegítimos siendo aceptados falsamente. Esto significa que el sistema tiene un nivel de seguridad alto.

- **Tasa de error equivalente (EER):** La EER es una métrica de rendimiento, un parámetro que se utiliza para medir y comparar el nivel de precisión general de diferentes métodos de autenticación biométrica. La EER corresponde al punto de intersección de dos curvas, FRR y FAR en función del umbral. Cuanto más bajos sean las FRR y FAR, más bajo será la EER, indicando un mejor rendimiento en la precisión del método de autentificación biométrica. En cierta literatura se utiliza como métrica de rendimiento la 'precisión', que por definición es la inversa de la EER.

$$EER = \frac{FAR+FRR}{2} \quad (2.6)$$

El rendimiento en términos de la precisión se puede visualizar gráficamente utilizando la curva de Característica Operativa del Receptor (ROC). Este gráfico se obtiene al trazar la tasa de aceptación verdadera (TAR) frente a la tasa de aceptación falsa (FAR) en diferentes valores de umbral. Siendo TAR la inversa de la tasa de rechazo falso ($1 - FRR$).

La probabilidad de falsa aceptación se puede reducir a costa de aumentar el falso rechazo. Por lo tanto, para cualquier sistema dado, el punto de operación se puede manipular para obtener una relación deseada entre falsas aceptaciones y falsos rechazos [9]. Todo el rango de posibles puntos de operación se caracteriza mediante la curva de Compensación de Errores de Detección (DET). La curva DET se obtiene al trazar la tasa de rechazo falso (FRR) frente a la tasa de aceptación falsa (FAR) en diferentes valores de umbral.

En la curva ROC, los diferentes clasificadores normalmente sólo difieren en la esquina superior izquierda del gráfico y parecen similares en una gran parte del mismo, debido a la escala lineal existente. En cambio, la curva DET suele presentarse utilizando un escalado de desviación normal, haciendo que los diferentes clasificadores presenten líneas rectas, y que el área de interés implique una gran parte del gráfico, facilitando así la distinción de los diferentes clasificadores según [36].

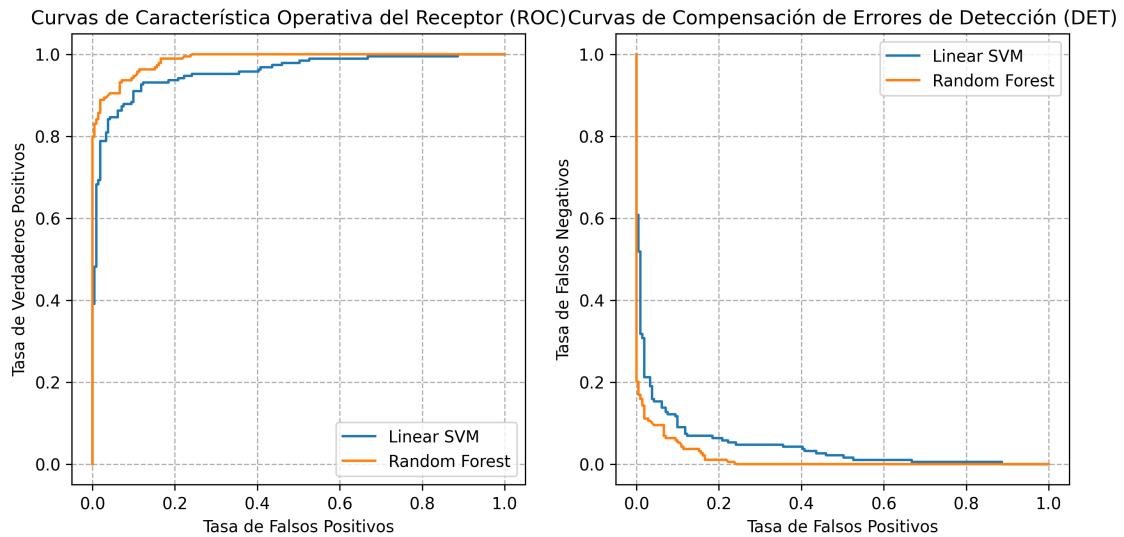


Figura 2.8: Comparación de las curvas ROC y DET utilizando diferentes algoritmos de clasificación

En la Figura 2.8 se ilustran las curvas ROC y DET de diferentes algoritmos de clasificación destinados a la misma tarea de clasificación. Un área mayor bajo la curva ROC (más cerca de la esquina superior izquierda del gráfico) indica un mejor rendimiento, mientras que un área menor bajo la curva DET indica un mejor rendimiento. Este comportamiento inverso se debe a la relación inversa existente entre la tasa de aceptación verdadera y la tasa de rechazo falso.

En la Figura 2.9 se ilustra la densidad de probabilidad de las puntuaciones 'scores' del locutor impostor (curva azul) y la densidad de probabilidad de las puntuaciones del locutor legítimo (curva naranja). Si se presenta un determinado umbral de decisión (línea negra vertical), la probabilidad de falsa aceptación corresponde con el área verde (área bajo la curva de densidad de probabilidad del locutor impostor en el intervalo $[umbral, x]$) y la probabilidad de falso rechazo corresponde con el área morada (área bajo la curva de densidad de probabilidad del locutor legítimo en el intervalo $[y, umbral]$). Donde x es la puntuación máxima del locutor impostor e y es la puntuación mínima del locutor legítimo.

Eficiencia del sistema

La eficiencia del sistema hace referencia al coste computacional impuesto por el método de autenticación biométrica. Cumplir con este criterio es especialmente crucial para dispositivos móviles que cuentan con recursos computacionales limitados. Un método de autenticación complejo puede provocar un mayor uso de recursos computacionales, lo que incrementa los tiempos de autenticación y disminuye la usabilidad del sistema [60]. Por eso, es esencial diseñar métodos de autenticación que minimicen la carga computacional.

Usabilidad del sistema

La usabilidad del sistema o aceptación del usuario en un sistema de autenticación es un factor importante para la implementación exitosa de un nuevo método de

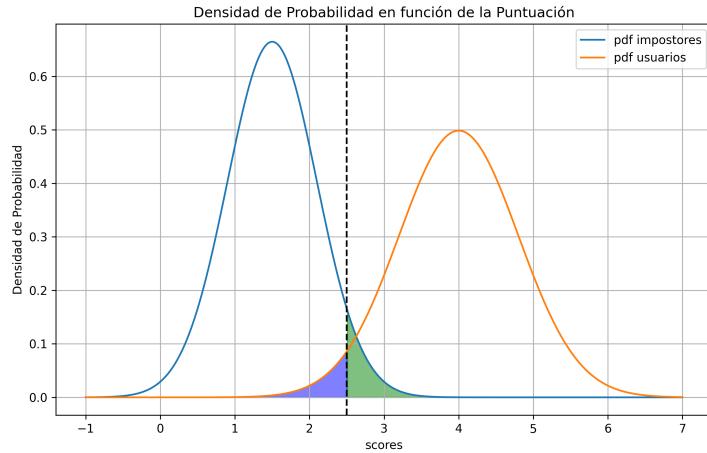


Figura 2.9: Funciones de densidad de probabilidad de las puntuaciones ‘scores’ de los locutores legítimos e impostores

autenticación. Los usuarios eventualmente abandonarán cualquier sistema tedioso o lento de usar, incluso si puede ofrecer un nivel más alto de protección de seguridad. Por lo tanto, un sistema de autenticación debe ofrecer un buen nivel de usabilidad del sistema, y esto se puede lograr, según [60]:

- Reduciendo la carga de trabajo impuesta al usuario lo máximo posible.
- Requeriendo la intervención del usuario lo menos posible.
- Haciendo que los retrasos de autenticación sean lo más cortos posible.

2.3. Síntesis de texto a voz (TTS)

Según [40], un sistema cuyo objetivo es convertir mensajes de texto ordinarios en un habla sintética inteligible y de sonido natural para transmitir información desde una máquina a un usuario humano, se denomina un sistema de síntesis de texto a voz (o TTS). En la Figura 2.10 se ilustra la estructura general de un sistema TTS, el cual realiza dos procesos fundamentales para generar la síntesis de voz:

- Módulo de análisis del texto que sirve para determinar la descripción lingüística subyacente abstracta del habla, es decir, pronunciación del texto, estructura sintáctica, enfoque semántico y resolución de ambigüedades.
- Sintetizador de audio, para generar la forma de onda del habla correspondiente al texto.

El origen de la síntesis de voz parte con la síntesis basada en reglas [34], posteriormente surge un modelo concatenativo [28], y un modelo basado en estructura paramétrica estadística [75].

Debido al avance de la IA, los modelos de síntesis de voz se han visto altamente mejorados [27]. Aunque, existen múltiples desafíos técnicos, por ejemplo, capturar con precisión los patrones prosódicos de la voz, duración y tono del habla. Para poder solventar estos desafíos técnicos, surge el primer modelo TTS extremo a extremo denominado Tacotron [69]. Al no depender de la parametrización estadística la generación del modelo, se ve mejorada la calidad del audio sintetizado. Aunque, la

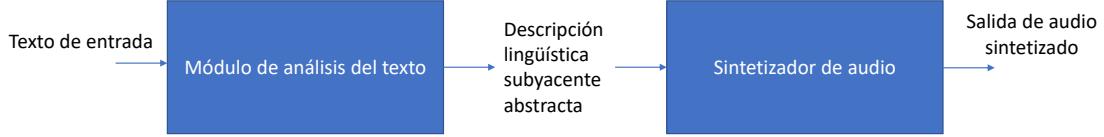


Figura 2.10: Diagrama de bloques general del sistema TTS

velocidad de inferencia era lenta, debido a la complejidad del modelo, por lo que no era funcional en aplicaciones de tiempo real. La baja velocidad de inferencia se soluciona utilizando FastSpeech.

FastSpeech está basado en Transformers, permitiendo computación paralela, debido al mecanismo de autoatención que tiene la arquitectura Transformer. Ante un texto de entrada al sistema TTS, se calculan las relaciones entre todas las palabras formantes de la secuencia de palabras entrante. Se realiza de manera independiente, para cada palabra en paralelo. Este procedimiento implica un tiempo de inferencia menor (velocidad de inferencia mayor). Además, la calidad del habla es mejor que respecto al modelo Tacotron, aunque se tenía poco control en el detalle del habla. Los patrones prosódicos seguían sin resolverse.

Los sistemas de síntesis de TTS se han vuelto tan poderosos que son capaces de generar voz realista, incluso si la base de datos asociada es limitada, esto implica que sea un peligro en sistemas de reconocimiento de locutor, debido a la naturalidad del habla generada.

El sistema de síntesis TTS utilizado en este trabajo se fundamenta en los modelos FastSpeech 2 y Tacotron 2; por consiguiente, se profundizará en dichos modelos en las siguientes secciones. Inicialmente, se explica Tacotron 2, que es una arquitectura seq2seq; luego, se presenta FastSpeech 2, que es un modelo no autorregresivo basado en Transformers, por lo que se explicará también la arquitectura del Transformer. Finalmente, se describen los vocoders utilizados, HiFi-GAN y DiffWave, los cuales son los encargados de generar las formas de onda a partir de los espectrogramas obtenidos por parte de Tacotron y FastSpeech.

2.3.1. Tacotron 2

Tacotron 2 es una arquitectura de red neuronal destinada a síntesis de voz, a partir de texto. Tacotron [69], se trata de una arquitectura de secuencia a secuencia (seq2seq) [58] que genera espectrogramas a partir de una secuencia de caracteres. Para generar la forma de onda a partir del espectrograma se utiliza el algoritmo Griffin-Lin [13], el cual establece una estimación de la fase del espectrograma. Esto es debido a que el espectrograma no tiene información de la fase, sino de amplitud asociada a cada una de las frecuencias en la ventana de tiempo concreta, por tanto,

es necesario para poder reconstruir la señal de audio, la amplitud, frecuencia (proporcionada por la transformada inversa de Fourier de tiempo reducido, (I-STFT)) y fase (proporcionada por el algoritmo de Griffin-Lin).

El modelo está formado por dos componentes principales:

1. **Red de predicción de características secuencia a secuencia, recurrente, con atención.** Predice una secuencia de fotogramas de Mel a partir de un texto en su entrada.
2. **Vocoder.** El vocoder toma el spectrograma predicho para formar la onda de voz correspondiente, se trata de una modificación de WaveNet [62], pero este vocoder no se utiliza en este trabajo, en cambio se utiliza uno denominado HiFi-GAN, el cual se explicará en secciones posteriores, Sección 2.3.3.

En la Figura 2.11 se muestra la arquitectura general de Tacotron 2. En color azul, se ilustra el esquema del codificador, formado por el embedder de caracteres, un bloque de capas de convolución y un bloque LSTM bidireccional; en color verde, la entrada correspondiente al codificador, que representa el texto de entrada; en color gris, el mecanismo de atención sensible a la ubicación, y en color naranja, el esquema del decodificador, formado por los bloques Pre-Net, Post-Net, LSTM unidireccional y proyección lineal. Finalmente, se obtiene el spectrograma Mel, a partir del cual, utilizando un vocoder, se genera el audio sintético.

A continuación, se explicará el spectrograma Mel, que es la representación final de la red de predicción, donde se plantea el porqué de su utilización y su caracterización. Posteriormente, se describe la arquitectura general de Tacotron 2, la cual está compuesta por un codificador de caracteres, una red de atención sensible a la ubicación y un decodificador para generar el spectrograma Mel.

2.3.1.1. Espectrograma Mel

Tanto la red de predicción como el vocoder se entrena a partir del spectrograma Mel. Esto hace que ambos componentes se puedan entrenar utilizando el criterio de minimización del error cuadrático (MSE), puesto que el spectrograma es invariante en fase dentro de cada cuadro. En cambio, si se utiliza la onda temporal, si es variante en fase y por tanto dificulta el entrenamiento mediante dicho criterio.

Al utilizar el spectrograma Mel, se aumenta la resolución en bajas frecuencias. En este espectro se plasma la inteligibilidad de la voz humana, y se disminuye la resolución en altas frecuencias, dominada por ráfagas de ruido y sonidos fricativos. Además, la pérdida de información de la fase se ve aumentada con respecto al spectrograma lineal, pero al utilizar el algoritmo Griffin-Lin, se pueden predecir las pérdidas de fase, permitiendo una correcta representación de la onda temporal, al utilizar la I-STFT.

El spectrograma Mel se obtiene utilizando la transforma de Fourier de tiempo reducido (STFT), con un tamaño de ventana y un salto de ventana determinados, de tal manera que sean ventanas pequeñas y superpuestas. Se utiliza una ventana “Hann”. La magnitud lineal dada por la STFT se transforma a escala Mel, utilizando un banco de filtros Mel en una compresión de rango dinámico logarítmico.

2.3.1.2. Arquitectura del modelo (Red de predicción del spectrograma)

La red de predicción está formada por un codificador y un decodificador con atención. El codificador tiene a su entrada un *embedding* de caracteres y a su salida

se obtiene una representación de características ocultas. El decodificador toma estas características y predice el espectrograma correspondiente, una representación intermedia del habla.

Codificador de caracteres

El codificador de caracteres sirve para obtener una representación de características ocultas de la secuencia de caracteres entrante. A continuación, se explica cómo se realiza dicha codificación:

- La secuencia de caracteres se representa en un *embedding* de caracteres de dimensión fija, generado por el embedder de caracteres.
- El *embedding* se pasa a través de un bloque de tres capas convolucionales unidimensionales (1D). Cada capa tiene un número determinado de filtros, donde cada filtro capta un conjunto específico de caracteres. Estas capas convolucionales modelan el contexto a largo plazo. Identifican y codifican relaciones entre los caracteres de entrada aunque se encuentren alejados entre sí.
- Al final de cada capa convolucional, se realiza una normalización por lotes [19], técnica que estabiliza y acelera el proceso de entrenamiento ajustando y escalando las activaciones para mantener la media y la varianza estables durante el entrenamiento. A continuación, se aplica una función de activación ReLU. Con esta función se introduce no linealidad al modelo y permite capturar relaciones más complejas en los datos.
- A la salida del bloque convolucional se presentan unos tensores, los cuales se conectan a una capa bidireccional de memoria larga a corto plazo (LSTM)² para así generar las características codificadas ocultas. Estas características ocultas son utilizadas por el mecanismo de atención para generar los vectores de contexto, los cuales serán utilizados por el Decoder para predecir el espectrograma Mel.

Red de atención sensible a la ubicación

En Tacotron 1 se utiliza una red de atención aditiva, en la cual, se presta atención a las diferentes partes de la secuencia de entrada para recopilar información importante en cada instante de tiempo. Sin embargo, en ciertos casos, el decodificador puede encontrar dificultades para avanzar sin problemas a través de la entrada, lo que da como resultado subsecuencias repetidas o ignoradas. Esto puede generar una síntesis de voz poco natural y distorsionada. Para abordar este problema, Tacotron 2 incorpora el mecanismo de atención sensible a la ubicación.

Las características ocultas se proyectan en la “red de atención” para crear un “vector de contexto”. Este vector resume la secuencia completa codificada para que sea útil en cada paso del decodificador. El mecanismo de atención considera la posición de cada elemento de la secuencia. Se trata de combinar la información de las diferentes posiciones, ponderada por su relevancia. El modelo recuerda qué

²La secuencia de entrada en una red neuronal recurrente (RNN) debe ser relativamente corta para que la memoria de la red tenga un efecto relevante en la predicción actual. Esto se resuelve utilizando redes de memoria larga a corto plazo (LSTM). Una red LSTM es capaz de recordar datos relevantes en la secuencia y preservarlos durante varios instantes de tiempo, permitiendo así manejar dependencias a largo plazo. Por tanto, a diferencia de las RNNs estándar, las redes LSTM tienen memoria tanto a corto como a largo plazo y además se resuelve el problema de desvanecimiento de gradiente que ocurre cuando la secuencia de entrada es larga en una RNN.

partes de la secuencia de entrada han sido atendidas, para decidir a qué partes prestar atención en el siguiente paso, para así distribuir mejor la atención a lo largo de la secuencia.

Las “características de ubicación” ayudan al modelo a conocer la posición relativa de los elementos de la secuencia. Se calculan mediante un filtro de convolución 1D. Lo que se hace es “enventanar” la secuencia de entrada en una longitud determinada y hacer una transformación lineal a cada subsecuencia. Al utilizar filtros diferentes, se captan diferentes parámetros de la secuencia. Estos parámetros convolucionados con las entradas proyectadas presentan la probabilidad de atención. Esta característica adicional ayuda a mitigar las posibles subsecuencias repetidas o ignoradas en el decodificador.

Decodificador de espectrograma Mel

El decodificador es una RNN autorregresiva, donde la atención decide qué datos provenientes del codificador se van a utilizar en el paso actual del decodificador.

Inicialmente, se proyecta la predicción del instante anterior en una pre-red. Esta pre-red está formada por diferentes capas totalmente conectadas, actúa como un cuello de botella de la información y conlleva un papel crucial en el aprendizaje de la atención. La salida de la pre-red y el vector de contexto de atención se concatenan y procesan mediante una pila de capas LSTM unidireccionales. Finalmente, se realiza una proyección lineal para obtener la predicción del espectrograma objetivo.

Además, se utiliza una red post-net para realizar un ajuste global de la predicción del espectrograma. Después de obtener la predicción total del espectrograma, se proyecta en una capa convolucional la cual predice un residual. Post-net toma el espectro Mel predicho por el decodificador y predice el residuo, el cual se agrega a la predicción de la pre-net para obtener la salida final.

Durante el entrenamiento de estos modelos autorregresivos, se emplea un enfoque de “teacher forcing”. En lugar de predecir toda la secuencia a la vez, se predice un cuadro a la vez, retroalimentando el modelo con la predicción del instante anterior para predecir el siguiente. Este proceso, conocido como seq2seq, implica que el decodificador recibe como entrada la predicción anterior durante el entrenamiento.

A continuación se detallan matemáticamente los componentes del mecanismo de atención en el contexto de un decodificador en RNNs, comúnmente utilizadas en modelos de secuencia a secuencia (seq2seq).

El vector de contexto c_i en el paso i del decodificador es una combinación ponderada de todos los vectores de características h_j , que son las salidas del codificador para cada posición j en la secuencia de entrada.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (2.7)$$

T_x es la longitud de la secuencia de entrada al modelo. La ponderación parte de los pesos de atención α_{ij} , los cuales son el resultado de normalizar las puntuaciones de atención e_{ij} para todas las posiciones j .

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^{T_x} e_{ik}} \quad (2.8)$$

La función de puntuación e_{ij} calcula la relevancia de cada vector de características

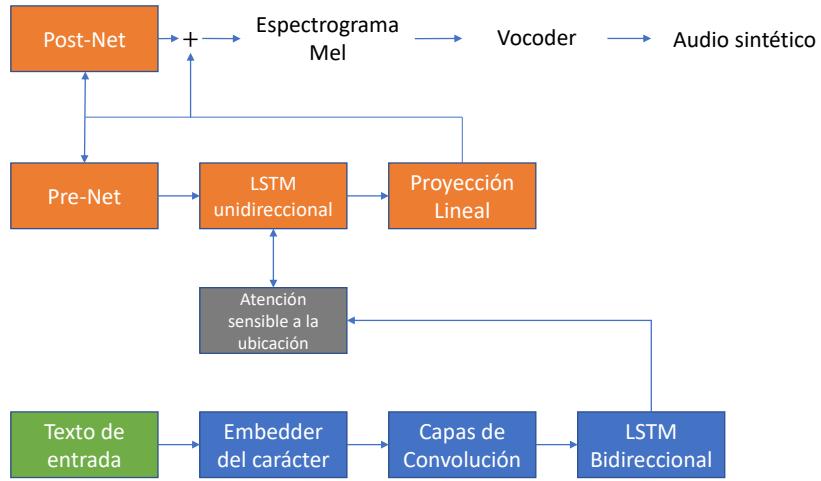


Figura 2.11: Arquitectura de Tacotron 2

h_j en relación con el estado previo del decodificador s_{i-1} .

$$e_{ij} = w^T \tanh(Ws_{i-1} + Vh_j + b) \quad (2.9)$$

e_{ij} es la puntuación de atención para el estado i y posición j , w es un vector de parámetros que transforma el resultado de la función de activación, b es el vector de sesgo, W es la matriz que transforma el estado previo del decodificador s_{i-1} y V es la matriz que transforma el vector de características h_j .

Se extiende la función de puntuación original utilizando el vector de ubicación f_{ij} , para así mejorar el mecanismo de atención, se incorpora información de la alineación previa α_{i-1} utilizando convolución. Por consiguiente, la función de puntuación queda plasmada de la siguiente manera:

$$e_{ij} = w^T \tanh(Ws_{i-1} + Vh_j + Uf_{ij} + b) \quad (2.10)$$

U es la matriz de parámetros que transforma el vector de ubicación f_{ij} .

El propósito de esta extensión es mejorar la atención debido a que el modelo tiene en cuenta las posiciones ya atendidas, de esta manera se puede evitar repeticiones y captar dependencias a largo plazo en la secuencia.

2.3.2. FastSpeech 2

Los modelos de síntesis de texto a voz (TTS) no autorregresivos (esto significa que el sistema no considera las muestras anteriores para la retroalimentación del sistema en su entrada), son más rápidos que los modelos autorregresivos, aunque la calidad en ambos sistemas es comparable. El entrenamiento del modelo FastSpeech se basa en un modelo maestro autorregresivo para la predicción de la duración y la destilación del conocimiento. Por una parte, se implementa información adicional a la entrada, y por otra parte se proporciona simplicidad en los resultados. Esto hace que el mapeo “uno a muchos” sea más eficiente. Aunque existen ciertas desventajas. Cuando se utiliza la destilación del conocimiento, existe un proceso de destilación maestro-estudiante, el cual es complicado y hace que tarde más en resolverse. Además, la duración del modelo maestro no es precisa y por tanto, los espectrogramas de Mel objetivo destilados del modelo maestro implican cierta pérdida de información, limitando así la calidad del modelo.

FastSpeech 2 surge para abordar tales desafíos y se realiza una mejor resolución del mapeo uno a muchos. Para ello, no se utiliza el resultado simplificado del maestro, sino que se introduce información de variación del habla como la duración, tono, más precisos. Se genera la forma de onda de voz a partir del texto, en paralelo. Por tanto, se tiene una conectividad total de extremo a extremo (sin depender de pasos intermedios), por consiguiente, se tiene una mayor eficiencia en la generación de voz, reducción de la complejidad, mejora de la calidad de la voz generada y una mayor flexibilidad. Se ha demostrado experimentalmente que FastSpeech 2 es mejor que FastSpeech, en términos de velocidad de inferencia, calidad de voz, e incluso supera a los modelos autorregresivos.

En la Figura 2.12 se muestra la arquitectura general del modelo FastSpeech 2. Como entrada, se implementa un texto que se divide en fonemas. A partir de estos fonemas, se generan los *embeddings* correspondientes, a los cuales se les aplica una codificación de posición antes de ingresarlos al codificador. El codificador produce una secuencia oculta de fonemas en su salida. Luego, el adaptador de varianza añade información sobre la duración, el tono y la energía a esta secuencia. Finalmente, el decodificador de espectrograma de Mel genera, en paralelo, una secuencia de espectrograma de Mel.

En el proceso de entrenamiento del modelo se utiliza el espectrograma Mel real como objetivo. Esto hace que se evite pérdida de información en dichos espectros, aumentando así la calidad de la voz sintetizada.

En los siguientes apartados se explicarán los diferentes componentes que conforman la arquitectura general de FastSpeech 2. Inicialmente, se detallará el embedder de fonemas, el módulo encargado de generar los *embeddings* de los fonemas. La codificación posicional se explicará en la sección 2.3.2. Luego, se describirán superficialmente el codificador y el decodificador de espectrograma Mel, los cuales están basados en la arquitectura Transformer. Consecuentemente, se explicará más en detalle el adaptador de varianza, compuesto por predictores de duración, tono (pitch) y energía. Finalmente, para comprender más en detalle como funciona el proceso de codificación y decodificación del espectrograma Mel se explica el modelo Transformer.



Figura 2.12: Arquitectura general de FastSpeech2

2.3.2.1. Embedder del fonema

A la entrada del modelo se presenta una secuencia de caracteres, esta secuencia se divide en palabras y cada una de estas palabras en sus fonemas correspondientes. Una capa de *embedding* entrenable, denominado embedder del fonema, transforma los fonemas en vectores (*embeddings*). Estos vectores tienen una dimensión que corresponde al tamaño del espacio de *embedding* de los fonemas.

2.3.2.2. Codificador y decodificador del espectrograma de Mel

La estructura del decodificador de espectrograma de Mel y del codificador se basa en un bloque de alimentación directa del Transformer (Feed-Forward Transformer Block, FFT Block), dicho bloque conforma parte del codificador del Transformer [64]. En Transformer, el bloque FFT está formado por dos capas, una de atención multi-head y una red de alimentación directa (FFN). Cada una de estas capas conlleva a su salida una adición de conexión residual y su correspondiente normalización. Estos componentes se explicarán en detalle en la sección 2.3.2.

2.3.2.3. Adaptador de varianza

Este módulo tiene el objetivo de agregar información adicional a la varianza de la secuencia oculta de fonemas, como la duración, energía y tono. Esto facilita el mapeo de uno a muchos en TTS.

La duración del fonema representa cuánto tiempo dura el fonema realmente, al realizar la contabilización de tiempo de todos los fonemas se obtiene la duración total de la sentencia. El tono es fundamental para transmitir emociones, por tanto, afecta a los parámetros prosódicos. La energía muestra la magnitud del espectrograma de Mel la cual afecta al volumen y al parámetro prosódico. Si se adjuntara más información de variación, se podría caracterizar más aún la voz a sintetizar.

En la Figura 2.13 se muestra la arquitectura del adaptador de varianza, el cual consta de un predictor de duración, tono y energía. Inicialmente, se realiza la predicción de la duración del fonema, a partir de esta se realiza una regulación de la longitud expandiendo los estados ocultos de la secuencia del fonema para que coincidan con la longitud de la secuencia del espectrograma Mel, permitiéndose así el control sobre la prosodia y la velocidad de la voz. Consecuentemente, se realiza la predicción del tono y de la energía. Además, se ilustra como es posible añadir otros predictores por si fuera necesario.

En la fase de entrenamiento, el adaptador de varianza trata de predecir la duración, tono y energía, con objetivo en la duración, tono y energía reales. En el proceso de inferencia, se utilizan los predictores finalmente hallados, para sintetizar la voz. Todos los predictores utilizan como optimización la minimización del MSE.

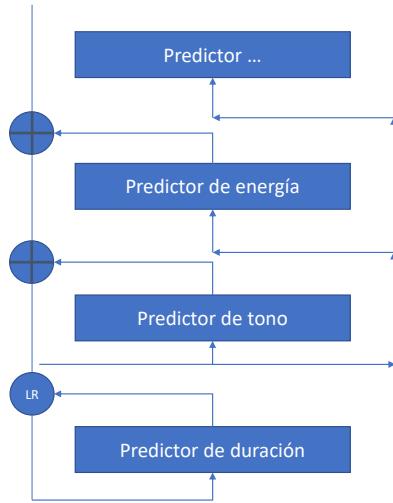


Figura 2.13: Arquitectura del adaptador de varianza

Predictor de duración

La entrada al predictor de duración es la secuencia oculta de fonemas proveniente del codificador y a su salida se extrae la duración de los espectrogramas Mel para cada fonema, para ello se utiliza una herramienta de alineación forzada de Montreal (MFA) [32]. Se trata de un algoritmo que toma un archivo de audio y su correspondiente transcripción y se calculan los instantes de tiempo correspondientes a cada palabra, fonema y fono que implica la transcripción [70]. El entrenamiento de este algoritmo se basa en modelos ocultos de Markov o Hidden Markov Models (HMMs)³. Debido a que la generación del espectrograma Mel objetivo es en paralelo, se presenta una alta velocidad en inferencia.

Predictor de pitch

El pitch real conlleva una variación alta, por tanto, cuando se estima con redes neuronales de predicción de pitch [11], la distribución de los valores de predicción de pitch es diferente a la distribución de pitch real. Como solución, se utiliza la transformada continua Wavelet (CWT). Al utilizar esta transformada se presenta información de frecuencia instantánea para cada uno de los puntos de la señal, además de ser ajustable y adaptable.

Un Wavelet es una forma de onda, cuya duración temporal es finita y su valor esperado es nulo. Su forma es irregular y asimétrica, permitiendo una mejor adaptación y convergencia para la señal a analizar [10]. Lo que se quiere es obtener el pitch de cada fonema de una forma precisa, utilizando esta transformada se puede analizar localmente temporalmente una señal, en esta se pueden apreciar aspectos que con una Transformada de Fourier Rápida (FFT), no se podría a simple vista, a no ser que se utilizara el enventanado correcto utilizando para ello la STFT.

³Según [41], HMM es una máquina estocástica de estados finitos capaz de producir a su salida una secuencia de símbolos observables. Se encargan de modelar de forma estadística la acústica de la señal de voz, es decir, cómo suenan los diferentes fonemas y palabras.

El contorno del pitch conlleva información sobre diferentes unidades lingüísticas en varias escalas temporales distintas. En una escala de tiempo pequeña, las señales micro prosódicas tienen una *naturaleza segmentaria*, y en escalas de tiempo normales, los tonos léxicos, acentos de las frases tienen *naturaleza lingüística*. Todo ello implica que existe una variación del pitch en diferentes escalas de tiempo. Es por ello que se utilizan wavelets para descomponer el contorno del pitch en cinco escalas temporales, que van desde la micro prosodia, hasta el nivel de enunciado. Cada componente escalada se entrena mediante un HMM. Una vez entrenadas las cinco componentes, se superponen en la etapa de síntesis.

En la Figura 2.14 se esquematiza la arquitectura del predictor del pitch. En el entrenamiento se extrae el contorno del pitch utilizando PyWorldVocoder¹⁰⁴. Mediante la transformada CWT se descompone el contorno del pitch en un espectrograma de pitch. Se toma este espectrograma de pitch como modelo de entrenamiento, este es el objetivo del predictor de pitch. En inferencia, se predice el espectrograma del pitch, y a este se le realiza la I-CWT para obtener el contorno del pitch.

Predictor de energía

Se calcula la norma L2 (raíz cuadrada de la suma de los cuadrados de los elementos del vector) a la amplitud asociada a cada trama del espectrograma. Se cuantifica uniformemente la energía de cada trama, y consecuentemente, se codifica en un *embedding* e , el cual se agrega a la secuencia oculta expandida.

En el entrenamiento, la predicción de la energía se basa en la energía real y no en el valor cuantificado. En este caso se utiliza transformada STFT, en vez de transformada CWT, puesto que la energía tiene una menor variación que el pitch.

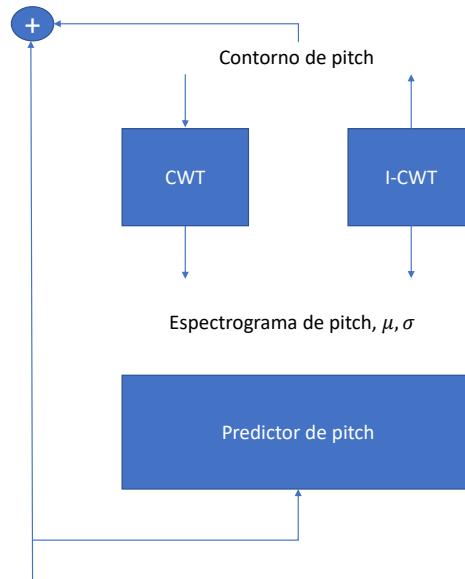


Figura 2.14: Esquema detallado del predictor del pitch.

⁴PyWorldVocoder10 es un software libre diseñado para el análisis, manipulación y síntesis de discurso de alta calidad. Este programa es capaz de estimar la frecuencia fundamental (F0), la aperiodicidad y la envolvente espectral [39]

2.3.2.4. Transformer

Un Transformer es una red neuronal de secuencia a secuencia (seq2seq), basada en mecanismos de atención, la cual prescinde de recurrencias y convoluciones. Transformer supera a muchos modelos basados en RNNs en traducción automática neuronal (NMT), por ejemplo. Esta red consta de dos componentes principales: un codificador y un decodificador.

Inicialmente, se va a explicar la motivación por la cuál se desarrolla esta red. Consecuentemente, se va a revisar el codificador, donde se explica cómo funciona cada uno de los componentes que lo conforma. Finalmente, se explica el decodificador, el cual conlleva muchos de los componentes que tiene el codificador, pero se encuentran ligeramente modificados y duplicados.

Motivación para el Desarrollo del Transformer

Una motivación para el desarrollo del modelo Transformer es que se quiere que la longitud de los caminos que deben atravesar las señales al aprender dependencias de largo alcance sea lo más corta posible. Esto es crucial si el modelo se basa en resumir un documento largo y el significado de una determinada palabra depende de algo encontrado varias oraciones antes. Un modelo recurrente, como una RNN estándar, que procesa palabras secuencialmente tiene longitudes de camino que crecen linealmente con la distancia entre dos posiciones de interés. En cambio, en un Transformer, las longitudes de camino siempre son constantes, independientemente de la distancia.

Otra motivación es que para secuencias largas, lo ideal es paralelizar el cálculo dentro de los ejemplos de entrenamiento. En el entrenamiento *mini-batch* se paralelizan los ejemplos, pero las limitaciones en memoria hacen que la paralelización dentro de los ejemplos sea deseable para acelerar los modelos recurrentes. Los modelos de secuencia convolucional alivian este problema, pero las longitudes de ruta son dependientes de la distancia. Los Transformers permiten una mayor paralelización que los modelos actuales y al mismo tiempo controlan dependencias de largo alcance. Aprovechan la auto-atención para calcular representaciones actualizadas para cada elemento de la secuencia en paralelo. El mecanismo de atención permite que cada representación considere de manera diferencial las representaciones de las demás posiciones y las rutas de comunicación tendrán la misma longitud para todos los pares de elementos.

Arquitectura del codificador

El codificador está compuesto por un número determinado N , de capas. Cada capa está formada por dos subcapas, Atención *Multi-head* y *Feed-Forward*, tal y como se ilustra en la Figura 2.15. La subcapa *Multi-head* se trata de un mecanismo de autoatención de múltiples cabezales, y la subcapa de alimentación directa, *Feed Forward*, se trata de una red de retroalimentación simple.

La entrada al codificador es una codificación de la secuencia entrante, puesto que la entrada inicial se segmenta en tokens (tokenización), cada token se codifica utilizando un *embedding* y cada *embedding* se codifica a su vez en función de la posición.

A continuación, se explican las diferentes capas que componen el codificador del Transformer, tal y como se muestra en la Figura 2.15. Primero, se detallan el

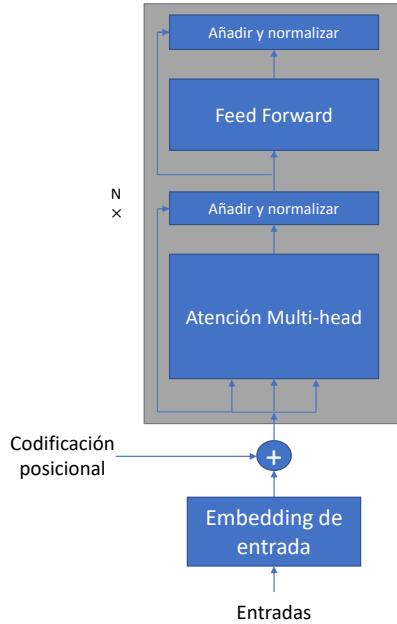


Figura 2.15: Codificador del Transformer

embedding de entrada y la codificación posicional, y luego se describen las subcapas que conforman la capa del codificador, incluyendo la atención, la atención multi-head, la subcapa de adición y normalización, y la red de alimentación directa (FFN).

- **Embedding de entrada:** Si se usa el lenguaje como dominio de ejemplo y se tiene como entrada una oración sin formato, según el vocabulario que se tenga, se asigna a cada palabra (token) un número entero único . El vocabulario estaría formado por decenas de miles de palabras. Estos tokens se pasan a través de una capa de generación de *embedding*, donde se tiene una tabla de búsqueda simple, la cual empareja cada entero con un vector continuo aprendido de alta dimensión. Este procedimiento es estándar en lenguaje natural.

Lo significante aquí es que la dirección del vector en este espacio de *embedding* puede corresponder con el significado semántico de la palabra asociada.

El objetivo de un Transformer es ajustar progresivamente estos *embeddings* para que no se limiten a codificar una palabra individual, sino que incorporen un significado contextual más amplio.

- **Codificación posicional:** En un modelo recurrente o convolucional , el orden de los elementos en una secuencia está implícito en la estructura misma de la red, pero en la atención no se procesa la entrada secuencialmente, por consiguiente, es necesario implementar información de posición en la codificación. Existe una tabla de búsqueda asociada a la posición de los *tokens*. Se propone un esquema de codificación fijo, donde la dimensión sigue una curva sinusoidal que disminuye en frecuencia a medida que el índice de dimensión i aumenta. Las dimensiones pares ($2i$) siguen una curva sinusoidal y las dimensiones impares ($2i + 1$) una curva cosenoidal, aunque estas curvas podrían

ser arbitrariamente diferentes. Si se toma el vector en la posición t y se realiza una codificación posicional en la posición $t + k$, para algunos desplazamientos fijos k , se puede calcular la codificación posicional como una función lineal de la codificación en posición t , es por ello que se escogen dichas curvas.

Las codificaciones posicionales tienen la misma dimensión que los *embeddings* de entrada, las cuales se suman antes de alimentar al codificador. La dimensión es d_{model} .

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad (2.11)$$

$$\text{PE}(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad (2.12)$$

i hace referencia a la dimensión, por lo que para cada posición pos y dimensión asociada i , se tiene una señal periódica a una determinada frecuencia.

- **Atención:** El primer paso del Transformer es la descomposición del texto en *tokens* y la asociación de cada *token* a un *embedding*. En este caso, el *embedding* es el mismo independientemente del contexto, ya que este *embedding* parte de una tabla de búsqueda sin referencia del contexto. En el siguiente paso del Transformer, los *embeddings* se codifican teniendo en cuenta la posición del token en la secuencia de caracteres entrante, y consecuentemente, adquieren significado contextual a partir de una función de atención (atención multi-head). El *embedding* entrante al mecanismo de atención describe qué *token* es y dónde se ubica en el texto y se denota como \vec{E}_i . Donde $i \in [1, N]$, siendo N el número de *tokens*.

Se desea que la mayoría de los cálculos se asemejen a productos de matriz-vector, donde las matrices están compuestas por pesos ajustables, los cuales varían durante el proceso de aprendizaje basándose en los datos. Estos parámetros se ajustan para minimizar alguna función de coste.

Si se tiene una único cabezal de atención, supóngase por ejemplo, que la tarea es que los adjetivos ajusten los significados de los sustantivos correspondientes. El objetivo es que una serie de cálculos produzca un nuevo conjunto refinado de *embeddings*, donde en este caso, los sustantivos hayan adquirido el significado de sus correspondientes adjetivos. El cabezal de atención tiene asociadas múltiples consultas, una consulta podría ser: ¿Existe algún adjetivo delante o detrás de mí? Si el texto es: “Una esponjosa criatura azul”, el sustantivo “criatura” hace dicha consulta y los adjetivos “esponjosa” y “azul” responden, por ejemplo.

Dicha consulta se codifica utilizando un vector numérico, denominado *Consulta (Query)*. El cálculo de la consulta consiste en tomar una matriz W_Q y multiplicarla por el *embedding* \vec{E}_i . El vector consulta resultante es \vec{Q}_i . Se multiplica W_Q con todos los *embeddings* del contexto \vec{E}_i , produciendo así un \vec{Q}_i para cada *token*.

Paralelamente, asociado a la consulta, existe una segunda matriz denominada *matriz Clave (Key)*, W_K . Esta matriz se multiplica con los diferentes *embeddings* produciendo una secuencia de vectores denominados *Keys*. Conceptualmente, las claves responden potencialmente a las consultas.

Tanto la matriz clave como la de consulta tienen parámetros ajustables y mapean los *embeddings* a un espacio dimensional más pequeño. Las claves coinciden con las consultas siempre que se alineen estrechamente entre sí. En el ejemplo, W_K asigna los adjetivos “esponjosa” y “azul” a vectores que están estrechamente alineados con la consulta producida por la palabra “criatura”. Para medir qué tan bien coincide cada clave con cada consulta, se calcula el producto escalar entre cada posible par clave-consulta. Los productos escalares más grandes corresponden a aquellos pares donde las claves y las consultas están más alineadas.

La cuadrícula de puntuaciones obtenida representa lo relevante que es cada palabra para actualizar el significado de todas las demás. La forma en la que se utilizan estas puntuaciones es tomando una suma ponderada a lo largo de cada columna, ponderada por su relevancia. Finalmente, se aplica *SoftMax* en todas las columnas. Esta cuadrícula se denomina “patrón de atención”, el cual se define como $A[i, j] = K_i \cdot Q_j$, mostrado en la Tabla 2.2. En el documento original, existe una forma compacta de describir esto, mostrada en la Ecuación 2.13.

Claves \ Consultas	Q1	Q2	Q3	Q4
K1	$\vec{Q}_1 \cdot \vec{K}_1$	$\vec{Q}_2 \cdot \vec{K}_1$	$\vec{Q}_3 \cdot \vec{K}_1$	$\vec{Q}_4 \cdot \vec{K}_1$
K2	$\vec{Q}_1 \cdot \vec{K}_2$	$\vec{Q}_2 \cdot \vec{K}_2$	$\vec{Q}_3 \cdot \vec{K}_2$	$\vec{Q}_4 \cdot \vec{K}_2$
K3	$\vec{Q}_1 \cdot \vec{K}_3$	$\vec{Q}_2 \cdot \vec{K}_3$	$\vec{Q}_3 \cdot \vec{K}_3$	$\vec{Q}_4 \cdot \vec{K}_3$
K4	$\vec{Q}_1 \cdot \vec{K}_4$	$\vec{Q}_2 \cdot \vec{K}_4$	$\vec{Q}_3 \cdot \vec{K}_4$	$\vec{Q}_4 \cdot \vec{K}_4$

Tabla 2.2: Patrón de atención del ejemplo propuesto

Se actualizan los *embeddings* para permitir que los tokens transmitan información a cualquier otro token para que sea relevante, es decir, que un token se desplace a una parte diferente de este espacio de los *embeddings* de alta dimensión que codifica más específicamente el token, relacionándolo con el contexto.

La forma más directa de hacer esto es utilizando una tercera matriz, denominada matriz de valores (*Values*), W_V . Al multiplicar W_V por los *embeddings* de los tokens, (\vec{E}_i) se obtiene una secuencia de “vectores de valores” y estos se añaden a los *embeddings*. Los vectores de valores están asociados a las claves correspondientes. En el ejemplo, al *embedding* de “criatura”, el sustantivo, se le añaden grandes proporciones de los vectores de valores de “azul” y “esponjosa”, debido a la consulta realizada. Los demás vectores de valor se reducen a cero.

La forma de actualizar el *embedding* es codificando previamente algún significado libre de contexto, produciendo un cambio que se quiera añadir denominado $\Delta(\vec{E}_i)$ y después se añade al *embedding* original, obteniéndose $\vec{E}'_i = \Delta(\vec{E}_i) + \vec{E}_i$. Esto resulta en un vector más refinado que codifica un significado más rico contextualmente. Este procedimiento se realiza en todos los *embeddings*.

$$A_t([Q], [V], [K]) = \text{Softmax} \left(\frac{[Q][K]^T}{\sqrt{d_k}} \right) [V] \quad (2.13)$$

Q y **K** representan las matrices completas de los vectores de consulta y clave, respectivamente. La expresión del numerador es una forma compacta de representar el patrón de atención, es decir, la red de todos los productos escalares posibles entre los pares $K_i - Q_j$. La matriz **V** es la matriz de valores. \mathbf{d}_k es la dimensión de la clave.

- **Atención “multi-head”:** Todo lo descrito anteriormente se denomina atención con un cabezal. La atención multi-head se basa en repetir múltiples veces un cabezal de atención. En el ejemplo central se han estudiado los adjetivos que actualizan los sustantivos, pero, hay muchas formas diferentes en el que el contexto puede influir en el significado de una palabra. Por ejemplo, “Ellos chocaron fuertemente el coche en un árbol”, como “chocaron” precede a “coche” tiene implicaciones en la forma y estructura del coche. Otras muchas asociaciones podrían ser menos gramaticales.

Para cada tipo diferente de actualización contextual, los parámetros de W_K y W_Q son diferentes para captar los diferentes patrones de atención y los parámetros del mapa de valores serían diferentes en función de lo que hubiera que añadir a los *embeddings*.

En el Transformer se ejecutan múltiples cabecezales de atención en paralelo, cada una con una atención diferente y por consiguiente, cada cabezal de atención tiene su propia consulta, claves y mapas de valores. Si se utiliza un número determinado N de cabecezales de atención, significa que existen N matrices (W_K y W_Q) diferentes, las cuales producen N patrones de atención diferentes. Como cada cabezal tiene asociada una matriz de valores, se producen N secuencias de vectores de valores. Todos ellos se suman utilizando como pesos los N patrones de atención. Esto se traduce en que cada uno de los cabecezales produce una propuesta de cambio, por tanto, se suman todos los cambios y se añade el resultado al *embedding* original en esa posición. Se muestra el procedimiento en la Ecuación 2.14.

$$\vec{E}'_i = \vec{E}_i + \Delta\vec{E}_{i1} + \Delta\vec{E}_{i2} + \Delta\vec{E}_{i3} + \dots \Delta\vec{E}_{iN} \quad (2.14)$$

- **Red de alimentación directa (FFN):** La red de alimentación directa (o Feed-Forward Network, FFN) o perceptrón multicapa es un componente crucial puesto que su objetivo principal es el de refinar los datos procesados por el mecanismo de atención.

Esta subcapa es fundamental para procesar los datos secuencialmente. Está construido como una red posicional totalmente conectada, esto significa que cada posición en la secuencia entrante se procesa por separado, pero equivalentemente, siendo crucial para mantener la integridad posicional de los datos de la entrada y asegurando uniformidad en la extracción de las características. La FFN está compuesta por dos capas lineales totalmente conectadas las cuales transforman los datos de entrada. Entre estas capas lineales se aplica una función ReLU, ayudando así a aprender patrones más complejos. En la Ecuación 2.15 se muestra matemáticamente la operación realizada por la FFN.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2.15)$$

W_1 y W_2 son las matrices de pesos asociadas a la primera y segunda capa, consecutivamente y b_1 y b_2 son los vectores de sesgo (bias) asociadas a la primera y segunda capa, consecutivamente. La función $\max(0, \cdot)$ representa la función de activación ReLU.

En esta subcapa los vectores no se comunican entre sí, sino que todos los vectores realizan la misma operación en paralelo. Se puede interpretar como si hiciera múltiples preguntas sobre cada vector y luego estos se actualizan en función de las respuestas.

- **Añadir y normalizar:** Después de cada subcapa (Atención y FFN) se aplica una conexión residual y una normalización de la capa. La conexión residual agrega una copia de la entrada a la salida, es decir, se agregan las representaciones de los tokens anteriores. Después de la normalización de la capa, toma los vectores de tokens y normaliza cada uno individualmente para que tenga media y varianza nula. Esto se propuso originalmente en RNNs para estabilizar el entrenamiento.

Arquitectura del decodificador

En esta sección se va a explicar qué es el enmascaramiento y la atención cruzada puesto que el decodificador sigue la misma estructura que el codificador, sin embargo, en vez de utilizar atención multi-head se utiliza atención multi-head con enmascaramiento y atención multi-head cruzada (atención encoder-decoder).

En la Figura 2.16 se muestra la arquitectura del decodificador del Transformer. Este decodificador recibe como entrada el texto objetivo desplazado a la derecha. Este texto objetivo retardado es primero transformado en vectores de *embeddings* que incluyen información posicional, representando así tanto el contenido como la posición de cada token. Luego, se aplica un mecanismo de atención con enmascaramiento a estos vectores de *embeddings*, lo que permite al modelo atender a las palabras anteriores en la secuencia sin acceder a las futuras. Despues, se realiza una atención cruzada que permite al decodificador concentrarse en diferentes partes del texto fuente codificado. Finalmente, los resultados pasan por una FFN. Tras cada mecanismo de atención y la capa FFN, se aplican capas de conexión residual y normalización.

- **Enmascaramiento:** Durante el proceso de entrenamiento, cuando se ejecuta el modelo Transformer en un ejemplo de texto dado y todos los pesos se ajustan y afinan ligeramente para premiar o sancionar en función de la probabilidad que asigne a la siguiente palabra verdadera del texto, resulta que el proceso de entrenamiento es más eficaz si se hace que se prediga simultáneamente cada posible siguiente token que siga a cada subsecuencia inicial de tokens de este texto. Es como si se estuvieran realizando múltiples entrenamientos.

A efectos del patrón de atención significa que nunca debe permitirse que los tokens posteriores influyan en los anteriores, por lo que se fuerzan a tener un valor nulo. Este proceso se denomina enmascaramiento y es relevante durante el proceso de entrenamiento. En conclusión, siempre se aplica este enmascaramiento para evitar que los tokens posteriores influyan en los anteriores.

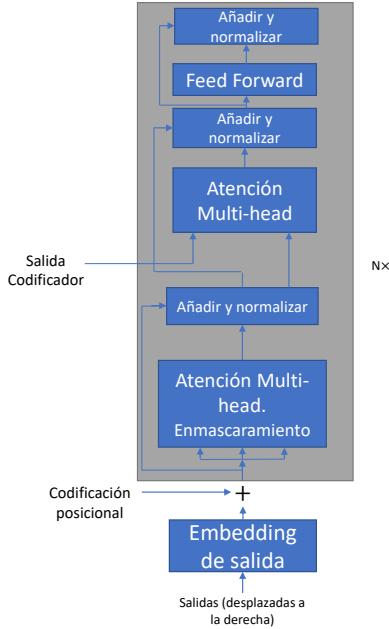


Figura 2.16: Decodificador del Transformer

- **Atención cruzada:** La atención cruzada implica modelos que procesan dos tipos diferentes de datos, como un sistema de traducción de texto, donde se presenta un idioma en la entrada y a la salida otro idioma diferente, o que la entrada sea audio y la salida sea su correspondiente transcripción textual. Un cabezal de atención cruzada tiene un aspecto casi idéntico a un cabezal de atención, la única diferencia es que los mapas de clave y consulta actúan sobre conjuntos de datos diferentes. En un modelo que realiza traducción, por ejemplo, las claves proceden de un idioma, mientras que las consultas de otro idioma y el patrón de atención podría describir qué palabras de un idioma origen corresponden a las palabras del idioma destino. En este escenario no suele haber enmascaramiento ya que no existe realmente la noción de que los tokens posteriores afecten a los anteriores, puesto que no se está realizando predicción.
En la atención vista hasta ahora, las claves, valores y consultas provienen de la misma secuencia, para el codificador esto es la oración fuente, pero para el decodificador es la oración objetivo. Los vectores no provienen todos de la misma secuencia, las consultas provienen de la capa anterior del decodificador y las claves y valores provienen de la salida final del codificador.
Durante la implementación esto permite que el decodificador preste atención a los tokens muestreados hasta el momento y también a la secuencia de entrada. Esto es crucial si se quiere que la salida realmente condicione la entrada, por ejemplo, que la frase de salida en francés corresponda a alguna entrada en inglés.
- **Salida del decodificador:** El resultado final de la última capa del decodificador es una representación vectorial para cada token en la secuencia objetivo. Para poder obtener la predicción se aplica una capa lineal para cada posición,

que asigna cada representación vectorial a un conjunto de lógicas no normalizadas sobre el vocabulario de salida, seguido de una función Softmax. Para establecer dependencia autorregresiva, el token se predice usando la salida del decodificador en el instante $t - 1$.

En el entrenamiento se utiliza como objetivo máxima verosimilitud ajustando los parámetros del Transformer para maximizar la probabilidad del token objetivo.

En función de la aplicación que se quiera puede no ser necesaria toda la arquitectura del Transformer, por ejemplo, si se quiere generar secuencias, una arquitectura de decodificador es suficiente. Si se requiere tareas de clasificación se podría hacer únicamente con el codificador.

2.3.3. Vocoder: HiFi-GAN

HiFi-Gan hace referencia a “Generative Adversarial Networks (GAN) for Efficient and High Fidelity (HiFi) Speech Synthesis”, redes generativas adversarias para síntesis de voz eficiente y de alta calidad.

En 2020 se realizó síntesis de voz utilizando redes generativas adversarias (GAN) para producir formas de onda sin procesar. Al utilizarse este tipo de redes, se mejora el muestreo y uso de memoria, pero la calidad es menor que en el caso de utilizar modelos generativos autorregresivos. Al mejorar el muestreo se capturan distribuciones complejas, generando así muestras realistas. HiFi-GAN modela los patrones periódicos de la voz para obtener una voz eficiente y de alta fidelidad. Sin embargo, modelar con precisión los patrones periódicos del habla sigue siendo un gran desafío debido a la naturaleza compleja de las señales sinusoidales con períodos variables.

La mayoría de los modelos de síntesis de voz neuronal utilizan dos etapas. Una etapa de predicción de baja resolución, por ejemplo, del espectrograma-Mel [35] o características lingüísticas [24], a partir de texto y una segunda etapa que consiste en sintetizar audio a partir de la representación intermedia. Por ejemplo, FastSpeech2 proporciona la primera etapa y la segunda etapa el vocoder HiFi-GAN.

En el contexto de la síntesis de voz, los vocoders basados en GAN tienen como objetivo generar formas de onda de voz realistas y con un sonido natural entrenando al generador para que produzca formas de onda de voz que puedan engañar al discriminador. El discriminador, por otro lado, está entrenado para distinguir entre formas de onda de voz reales y generadas. HiFi – GAN es un conjunto de redes conformadas por un generador y dos discriminadores. Discriminadores de múltiples escalas y períodos. El generador y discriminadores se entranan adversariamente, añadiéndole dos perdidas adicionales para mejorar la estabilidad del entrenamiento.

En la Figura 2.17 se muestra la arquitectura del vocoder HiFi-GAN, en ella se observa en color azul el generador y los discriminadores de múltiples períodos y escalas. En color gris las pérdidas del espectrograma Mel, GAN y de emparejamiento de características. Existen dos entradas, el espectrograma Mel y el audio original.

2.3.3.1. Generador

La red generadora de un vocoder basado en GAN normalmente toma como entrada algún tipo de características acústicas de bajo nivel, como espectrogramas Mel o parámetros del vocoder, y las transforma en formas de onda de voz. El generador

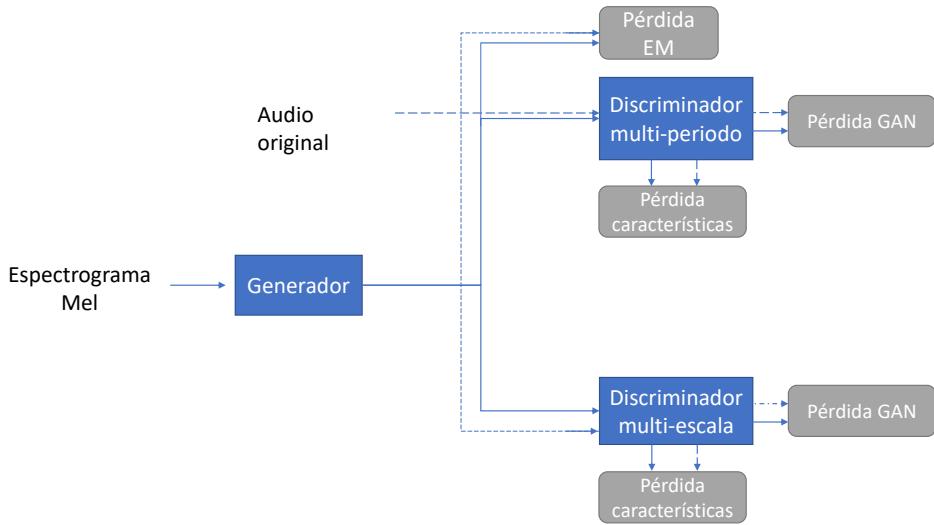


Figura 2.17: Arquitectura del modelo HiFi-GAN

aprende a producir formas de onda de alta calidad que se asemejan al habla real mediante la optimización de sus parámetros a través de un proceso de entrenamiento contradictorio.

En HiFi-GAN el generador es una red neuronal convolucional. A la entrada del generador se tiene el espectrograma de Mel y a su salida una secuencia con una longitud equivalente a la resolución temporal de las formas de onda sin procesar, debido a que se realiza un número de convoluciones traspuestas tal que se obtenga la longitud deseada, debido a que la convolución traspuesta implica sobremuestreo.

A la salida de cada convolución se tiene un módulo de fusión de campo multireceptivo (MRF). Este módulo hace que el generador observe patrones de diferentes longitudes simultáneamente. Este módulo está compuesto por múltiples bloques residuales, donde cada bloque tiene un determinado tamaño de Kernel y velocidad de dilatación para así formar diversos patrones.

2.3.3.2. Discriminador

La red discriminadora suele ser un clasificador binario, está entrenada para diferenciar entre las formas de onda del habla real y las formas de onda sintetizadas generadas por el generador. Aprende a identificar las características del habla real y a proporcionar retroalimentación al generador, guiándolo para que produzca una salida más realista.

El discriminador en HiFi-GAN está compuesto por dos componentes diferentes, donde cada componente está compuesto por múltiples sub-discriminadores.

- **Discriminador de múltiples periodos (MPD):** El MPD está formado por múltiples sub-discriminadores, cada uno diseñado para reconocer una porción distinta de las señales periódicas presentes en el audio. Al examinar diferentes segmentos de audio se capturan los patrones periódicos, identificando así dependencias a largo plazo.
 - Procedimiento: Se transforma la señal de audio unidimensional en una

matriz bidimensional en función del periodo de operación p del subdiscriminador, la matriz tiene dimensiones $(p, T/p)$. A esta matriz se le aplica convolución 2D para obtener las características de la señal.

- **Discriminador de múltiples escalas (MSD):** El MSD está diseñado para capturar patrones consecutivos, dependencias a largo plazo en la secuencia de audio y detalles finos, esto se debe a que al evaluar el audio en múltiples escalas se analiza completamente.
 - Procedimiento: Se segmenta la señal de audio en segmentos de N muestras consecutivas, se realiza la media a cada segmento, y al resultado se le aplica una convolución 1D para capturar las diferentes características de la señal. Se trata de una especie de submuestreo.

Al combinar MPD y MSD se captura de manera efectiva los patrones periódicos y dependencias a largo plazo del audio, permitiendo así una síntesis de voz de alta calidad. Se propone en [26] dicho discriminador, el cual se basa en analizar consecutivamente muestras de audio en diferentes escalas.

2.3.3.3. Funciones de pérdida en el entrenamiento

A continuación, se describen las tres funciones de pérdidas utilizadas en el entrenamiento de esta red adversaria. Una es intrínseca a esta red, las otras dos se implementan para incrementar la calidad y eficiencia del modelo entrenado.

Pérdida GAN

El generador y el discriminador tienen como objetivo de entrenamiento la minimización del error cuadrático. El discriminador se entrena para clasificar las muestras reales x en 1 y las muestras sintetizadas provenientes del generador $G(s)$ en 0.

Las funciones de pérdida adversariales se definen de la siguiente manera:

$$L_{\text{adv}}(D) = E \left\{ (D(x) - 1)^2 + (D(G(s)))^2 \right\} \quad (2.16)$$

$$L_{\text{adv}}(G) = E \left\{ (D(G(s)) - 1)^2 \right\} \quad (2.17)$$

- $L_{\text{adv}}(D)$ es la función de pérdida adversarial del discriminador. Mide qué tan bien es capaz de distinguir el discriminador entre las muestras reales x y las muestras sintetizadas por el generador $G(s)$. $D()$ es la clasificación que plantea el discriminador, cuya salida es “0” o “1”. $D(x)$ es la salida del discriminador frente a la muestra real x . $D(G(s))$ es la salida del discriminador frente a la muestra sintetizada por el generador $G(s)$.
- $L_{\text{adv}}(G)$ es la función de pérdida adversarial del generador. Mide qué tan bien el generador es capaz de “engaños” al discriminador.

E es la esperanza matemática, esta operación garantiza que la pérdida se calcule de manera adecuada para todas las muestras en el conjunto de datos.

Pérdida del espectrograma de Mel

Se implementa una pérdida adicional referente al espectrograma de Mel, para mejorar la etapa de entrenamiento y la fidelidad del audio sintetizado [74].

La pérdida del espectrograma de Mel es la distancia L_1 entre el espectrograma Mel de una forma de onda sintetizada por el generador para una condición de entrada s , $\phi(G(s))$ y el espectrograma Mel de la forma de onda de referencia $\phi(x)$.

La pérdida del spectrograma de Mel, $L_{\text{Mel}}(G)$ se define como:

$$L_{\text{Mel}}(G) = E \|\phi(x) - \phi(G(s))\|_1 \quad (2.18)$$

- $\phi(\cdot)$ es la función de transformación para obtener el spectrograma Mel.
- $\|\cdot\|_1$ es la norma L_1 , se calcula como la suma del valor absoluto de los elementos del vector.

Pérdida de emparejamiento de características

Al implementar la pérdida de emparejamiento de características, el generador produce características de activación similares en el discriminador para muestras de referencia y generadas.

Para calcular la pérdida de emparejamiento de características, se extraen las muestras de referencia x y las muestras generadas $G(s)$. Cada capa del discriminador produce características a diferentes niveles de abstracción. Se calcula la distancia L_1 entre las características intermedias de las muestras de referencia $D_i(x)$ y las muestras generadas $D_i(G(s))$ en cada capa del discriminador.

La pérdida de emparejamiento de características, $L_{\text{FM}}(G)$ se define como:

$$L_{\text{FM}}(G) = E \left\{ \frac{1}{T} \sum_{i=1}^T \frac{1}{N_i} \|D_i(x) - D_i(G(s))\|_1 \right\} \quad (2.19)$$

- T es el número de capas en el discriminador.
- N_i es el número de características en la i -ésima capa del discriminador.
- $D_i(\cdot)$ representa las características intermedias en la i -ésima capa del discriminador.

2.3.4. Vocoder: DiffWave

DiffWave es un modelo probabilístico de difusión destinado a generar formas de onda condicionales e incondicionales. Está basado en una señal de ruido blanco, la cual se convierte en una forma de onda a partir de una cadena de Markov.

Diffwave puede generar formas de onda a partir del spectrograma Mel. Se puede implementar una condición de clase, por ejemplo, para que las formas de onda generadas sean referentes a voz humana, instrumento musical, efecto de sonido, etc. Además, se puede generar forma de onda de manera incondicional, esto quiere decir que no existe ninguna entrada específica para la generación de audio, generando por consiguiente sonidos de manera espontánea y variada, manteniendo una alta fidelidad en la calidad de audio. Diffwave supera a los modelos basados GAN en términos de generación incondicional, ya que presenta una mejor calidad de audio y diversidad de muestras.

En Diffwave se utiliza un modelo probabilístico de difusión, se trata de una clase de modelo generativo que utiliza una cadena de Markov para convertir gradualmente una función de densidad de probabilidad (fdp) simple en una fdp compleja [18]. A diferencia que en GAN, no se requieren redes neuronales adicionales para el entrenamiento, evitando así los problemas de “colapso posterior” o “colapso de modo” producidos en el entrenamiento conjunto de dos redes, como en el generador y discriminador en GAN.

A continuación, se explica cómo la densidad de probabilidad de una distribución puede modelar los datos de síntesis. Consecuentemente, se explica la teoría referente a los modelos de difusión, para ello se explica el proceso *Forward*, proceso *Reverse* y cómo se realiza el entrenamiento de dichos modelos.

2.3.4.1. Modelado de la densidad de probabilidad para la síntesis de datos

En aprendizaje automático (ML) se asume que los datos provienen de una distribución, independientemente del tipo de dato, como imagen, vídeo, sonido, etc. Se asume que tiene una distribución subyacente denominada $p_{\text{datos}}(x)$ y que el conjunto de datos x , son muestras de dicha distribución. El objetivo de los modelos generativos de aprendizaje automático (GML) es básicamente aprender la distribución para después generar nuevos datos muestreando la distribución aprendida $p_{\theta}(x)$.

La distribución no es conocida y se aprende en el entrenamiento a partir de las muestras de datos reales, esto se puede realizar mediante dos métodos. El modelo se entrena minimizando la divergencia entre $p_{\text{datos}}(x)$ y $p_{\theta}(x)$. Si para una muestra el valor de probabilidad es alto, entonces existe una alta probabilidad de que la muestra provenga de la distribución. Lo que se intenta es maximizar la probabilidad de las muestras que el modelo genera. Otra forma es minimizar una métrica de divergencia entre la distribución de datos y la aprendida por el modelo. Esto se aplica en redes generativas, como las GAN, donde se tiene una función de pérdida adversarial.

El procedimiento en el que se basa GAN, por ejemplo, es que se parte de una muestra de ruido en la entrada del modelo y a su salida se obtiene una muestra que parece provenir de la distribución de los datos. En los modelos de difusión se realizan múltiples pasos explícitos para llegar hasta la estimación de la distribución de datos, de esta manera es más fácil de analizar el modelo. Cada uno de los pasos son cadenas de Markov, las cuales se basan en ecuaciones diferenciales estocásticas.

La principal idea detrás de la difusión es convertir una distribución base, bien conocida y simple, como por ejemplo una distribución Gaussiana, a una distribución *target*, de forma iterativa, utilizando cadenas de Markov.

2.3.4.2. Arquitectura del modelo de difusión

Los modelos probabilísticos de difusión son modelos generativos que implican dos procesos principales:

- **Proceso de avance (*Forward*)**

Consiste en tomar una muestra, por ejemplo, una imagen, x_0 y agregarle ruido gradualmente, en un número determinado de iteraciones, T . El ruido agregado se parametriza mediante una distribución Gaussiana. En cada iteración t , se tiene un parámetro denominado β_t . El conjunto de valores de β_t se denomina “programa de ruido”, puesto que controla cuánto ruido se agrega en cada iteración.

Si se presenta un número determinado de iteraciones T de adición de ruido, se demuestra matemáticamente que si T tiende a infinito, la muestra original se considera como una distribución Gaussiana, por ejemplo, ruido aleatorio. Se demuestra que x_t se puede expresar en términos de x_{t-1} como una función Gaussiana con distribución $q(x_t|x_{t-1})$. Como este proceso se define de manera recursiva, se puede expresar x_t en términos de x_0 como una distribución Gaussiana, $q(x_t|x_0)$.

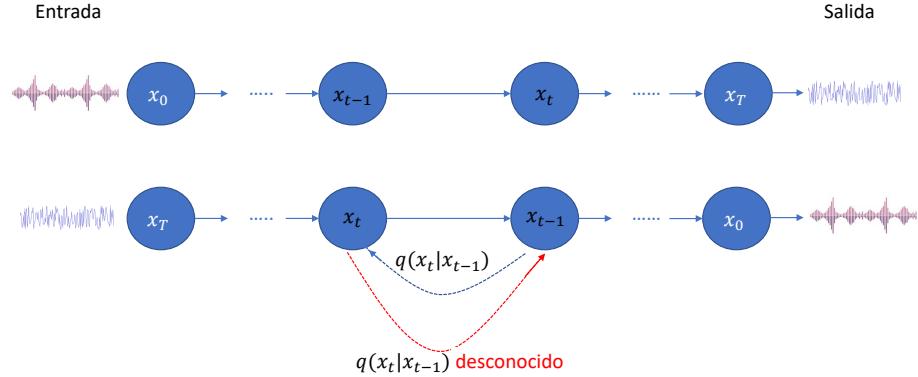


Figura 2.18: Arquitectura del modelo de difusión. Proceso Forward (parte superior) y proceso Reverse (parte inferior)

■ Proceso inverso (*Reverse*)

En GAN se demuestra cómo se obtiene el audio a partir de ruido, y esto es lo que el proceso *Reverse* realiza. Se parte de ruido e iterativamente se va reduciendo este para así poder obtener el audio original. El objetivo de un modelo de difusión es aprender cómo hacer el proceso *Reverse*. En este caso, lo que se quiere es conocer $q(x_{t-1}|x_t)$, es decir, la distribución inversa de $q(x_t|x_{t-1})$. Conocer esta distribución es complejo, por lo que se calcula una aproximación de dicha distribución estimando que $q(x_{t-1}|x_t)$ es una distribución Gaussiana también. Por consiguiente, se puede parametrizar el proceso *reverse* aprendido como una distribución normal $p_\theta(x_{t-1}|x_t)$.

Una función Gaussiana se parametriza con dos términos: la media y la varianza. Por tanto, la difusión intenta aprender estos dos términos a través de una red neuronal.

En la Figura 2.18 se muestran esquemáticamente los procesos anteriormente explicados. En el proceso Forward se observan las T iteraciones en las cuales se va añadiendo gradualmente el ruido. Se parte de una muestra de audio real, x_0 y se obtiene a la salida de la última cadena de Markov una muestra de ruido, x_T . En el proceso Reverse se parte de una muestra de ruido, x_T e iterativamente se va reduciendo el ruido hasta obtener la muestra de audio original x_0 .

2.3.4.3. Entrenamiento del modelo de difusión

Para poder entrenar el modelo es necesario una función de pérdida. Se maximiza la probabilidad de que el audio que se genera parezca provenir de la distribución de datos original, pero esa probabilidad no se puede calcular. Sin embargo, se puede limitar la probabilidad utilizando la variación del límite inferior. Se trata de minimizar la pérdida L_2 entre la media de la distribución aprendida y la media de la distribución condicional, de esta manera, se intenta predecir el ruido que se agregó.

A partir de un audio ruidoso, se le resta el ruido añadido en el proceso *Forward* para recuperar el audio original. Pero, si se comienza con ruido puro y se le resta algún ruido, es posible que se obtenga un audio nuevo. Esta es la intuición principal detrás de los modelos de difusión.

- **Algoritmo de entrenamiento**

Se tiene un audio x_0 , con una distribución de probabilidad asociada $q(x_0)$. Se agrega ruido, no iterativamente en $t = 1, 2, \dots, T$, sino en una única iteración, obteniéndose x_T . Si se tiene en la entrada del modelo x_T , el modelo predice el ruido agregado a x_0 y se intenta minimizar la distancia entre el ruido real y el estimado. Dicho algoritmo se repite hasta que el modelo converja. Se trata de un bucle de entrenamiento estándar.

Una vez que el modelo se ha entrenado, la forma para poder sintetizar nuevo audio es comenzando con ruido aleatorio, como si se tratara de un audio ruidoso. Este se pasa al modelo prediciendo algún ruido de dicha entrada; se realiza la diferencia del audio ruidoso con el ruido estimado, obteniéndose así un audio sin ruido.

Capítulo 3

Estudio experimental de los sistemas TTS y ASV analizados teóricamente

3.1. Introducción

En la sección 2.2.4 se explica que, para poder evaluar correctamente un sistema de verificación automática de locutor (ASV) ante ataques de suplantación (spoofing), es necesario evaluar el sistema en dos escenarios:

- Utilizar como entradas al sistema de ASV el locutor objetivo (LO) y el locutor impostor (LI), tratándose de una evaluación del rendimiento del sistema de referencia (o estándar).
- Utilizar como entradas al sistema de ASV el LO y el locutor objetivo sintetizado (LOS), tratándose de una evaluación del rendimiento del sistema ante suplantación.

Es por ello que el estudio de los sistemas de ASV utilizados experimentalmente se realizarán en estos dos escenarios. Inicialmente, se analizará la precisión del sistema de ASV estándar; luego, se analizará la precisión del sistema de ASV en condiciones de suplantación. La diferencia del rendimiento entre ellos refleja la vulnerabilidad del sistema al ataque de suplantación en particular considerado. La precisión se calculará a partir de la tasa de error equivalente (EER), siendo esta por definición la inversa de la precisión o exactitud del sistema, según [60].

3.2. Estudio del rendimiento de los sistemas ASV utilizados

Inicialmente, se realiza el estudio del rendimiento de referencia de los sistemas de verificación automática de locutor (ASV) analizados teóricamente. Para ello, se opta por analizar a 20 locutores objetivos (10 masculinos y 10 femeninos), los cuales se inscriben en el sistema de ASV correspondiente. Cada uno de los locutores inscritos se confronta con respecto a 20 locutores impostores (10 masculinos y 10 femeninos).

Para cada locutor objetivo, se obtiene un *embedding prototipo* asociado, correspondiente a la inscripción (o 'enrollment') de dicho locutor en el sistema ASV. Se

trata del modelo que guarda el sistema de ASV para verificar la identidad del locutor en la fase de evaluación.

En este estudio se utilizan tres sistemas de ASV diferentes. El sistema x-vector, descrito en 2.2.3, se basa en una estructura codificadora TDNN, una particularización de la DNN. El sistema ResNet, descrito en 2.2.3, donde la estructura codificadora TDNN se sustituye por una red neuronal residual 2D y el sistema ECAPA-TDNN, descrito en 2.2.3, el cual es una mejora de x-vector y ResNet.

3.2.1. Caracterización del locutor objetivo

La caracterización del locutor objetivo se basa en generar 20 *embeddings prototipo*, para generar 20 inscripciones de 20 locutores objetivo (LOs). Estos diferentes locutores parten de la base de datos LibriTTS¹.

Para lograr una inscripción correcta del locutor objetivo en el modelo de ASV, se extraen múltiples *embeddings* del locutor objetivo, cada uno asociado a una sentencia diferente. Estos *embeddings*, al ser vectores numéricos, pueden analizarse para determinar una representación característica del locutor. Si se comparan múltiples *embeddings* del mismo locutor, las puntuaciones resultantes conllevan una distribución aproximadamente Gaussiana, por consecuencia, una medida representativa adecuada es la media estadística de todos los *embeddings* obtenidos. Esta media de los embeddings obtenidos se presenta como el *embedding prototipo*. El objetivo es que el *embedding prototipo* refleje las características del locutor de manera independiente al contenido del texto, permitiendo definir correctamente al locutor objetivo sin importar lo que diga.

Es necesario realizar una inscripción completa para que el sistema verificador sea robusto. Esto se puede hacer utilizando múltiples sentencias. Por lo tanto, es necesario que el LO disponga de una gran cantidad de sentencias, es por ello que se realiza una automatización para hallar el número de archivos de audio que contiene cada uno de los locutores que conforma dicha base de datos y ordenar los locutores de mayor a menor cantidad de audios (este algoritmo también se utiliza en la caracterización del locutor impostor). Como respuesta se obtienen los locutores mostrados en la Tablas 3.1 y 3.2. Las tablas muestran el identificador del LO y el número de sentencias asociadas. Utilizando estas tablas se puede establecer el número de sentencias que se podrían asociar al cálculo del *embedding prototipo* y el número de sentencias que se podrían utilizar para realizar la comparación con el *embedding prototipo*. Para así obtener las puntuaciones (scores), las cuáles son uno de los datos base que proporciona el rendimiento y robustez del sistema de ASV.

Destacar que las sentencias que se utilizan para generar el *embedding prototipo* son diferentes a las sentencias utilizadas para generar las puntuaciones asociadas. De esta manera el análisis es más exhaustivo.

Se deduce a partir de las Tablas 3.1, 3.2 que el número de archivos de audio máximo es 815, mientras que el número mínimo de archivos de audio es 387, por tanto,

¹LibriTTS es un corpus de inglés para varios locutores de aproximadamente 585 horas de lectura de voz en inglés a una frecuencia de muestreo f_s de 24 kHz. Fue constituido por Heiga Zen con la ayuda de los miembros del equipo de Google Speech y Google Brain. El corpus LibriTTS está diseñado para la investigación de TTS. Se deriva de los materiales originales (archivos de audio mp3 de LibriVox y archivos de texto del Proyecto Gutenberg) del corpus LibriSpeech.

ID LO-Masculino	Número Sentencias
5181	815
5198	556
5328	510
978	453
3962	435
3433	425
6539	421
2309	418
4442	411
7423	387

Tabla 3.1: Locutores Objetivos Masculinos de la base de datos LibriTTS

ID LO-Femenino	Número Sentencias
2834	511
614	506
585	470
483	444
5949	443
215	436
6127	429
1384	424
3867	407
4779	395

Tabla 3.2: Locutores Objetivos Femenino de la base de datos LibriTTS

se van a utilizar 100 sentencias para generar el *embedding prototipo* asociado a cada uno de los LOs y se van a utilizar 200 sentencias por cada LO para la evaluación del sistema de ASV. Al representar las puntuaciones obtenidas para los 20 locutores diferentes, se plasma el comportamiento general del sistema de ASV cuando la persona que habla es quien dice ser, independientemente del sexo del locutor.

3.2.2. Caracterización del locutor impostor

Para realizar la caracterización del LI en el sistema ASV se va a utilizar como base de datos, VoxCeleb 2². Se buscan los 10 locutores impostores (LIs) masculinos y 10 LIs femeninos con la máxima cantidad de sentencias asociadas en la base de datos VoxCeleb 2, por ser mayor que la base de datos VoxCeleb 1. Obteniéndose los LIs masculinos con identificadores: 15,29,60,62,168,184,185,191,255,258 y los LIs femeninos con identificadores: 43,149,332,385,650,673,680,755,806,829, donde todos los LIs tienen 500 sentencias asociadas.

Cada sentencia generará un *embedding* asociado cuando se implemente en inferencia, en el sistema de ASV utilizado: x-vector (TDNN), ResNet o ECAPA-TDNN. Este *embedding*, al compararse con el *embedding prototipo* del LO, generará una puntuación asociada utilizando la distancia coseno. Mediante esta puntuación se caracteriza al LI en el sistema de ASV.

3.2.3. Métodos

En esta parte del trabajo se realiza el análisis del rendimiento de referencia de los sistemas de ASV basados en x-vector, ResNet y ECAPA-TDNN, para realizar este estudio es necesario establecer una serie de métodos, principalmente basados en el histograma de puntuaciones de los diferentes locutores de entrada.

En la sección 2.2.4 se demuestra que la precisión de la verificación del locutor se establece mediante la tasa de error equivalente (EER), y que para poder calcularla es necesario contabilizar la tasa de falsos negativos (FNR) y la tasa de falsos positivos (FPR). Es posible obtener la EER gráficamente de diferentes maneras, en este caso

²VoxCeleb 2 es una base de datos que contiene más de 1 millón de sentencias de 6112 celebridades, extraídas de videos subidos a YouTube

se halla a partir de la gráfica conjunta de tasa de falsos positivos y tasa de falsos negativos en función del umbral de puntuación definido.

Para representar eficientemente los resultados (puntuaciones del ASV), se ha utilizado un *histograma de las puntuaciones* dadas para los locutores objetivos (LOs) y los locutores impostores (LIs). Para observar la precisión del sistema de ASV, se muestra un *histograma de puntuaciones conjunto* que incluye los LOs y los LIs. De manera similar, para caracterizar la precisión del sistema de ASV según el sexo, se presenta un *histograma de puntuaciones conjunto* de LOs y LIs masculinos por un lado, y de LOs y LIs femeninos por otro lado.

3.2.3.1. Histograma de puntuaciones del locutor objetivo agregado

El histograma de puntuaciones del LO se calcula a partir de 200 puntuaciones para cada LO. Como se dispone de 20 LOs, se presentará un total de 4000 puntuaciones para dicha distribución. Cada puntuación asociada a cada sentencia viene dada por la distancia coseno entre el *embedding* obtenido para la sentencia asociada de dicho LO y el *embedding prototipo* del LO. Cuanto más cercana sea la puntuación a 1, más probable será que el locutor de prueba coincida con el locutor objetivo y cuanto más cercana sea la puntuación a 0, menos probable será que el locutor de prueba coincida con el locutor objetivo, lo que indicaría que es un impostor. En este primer análisis es razonable decir que todas las puntuaciones se ubicarán cercanas a la unidad, puesto que el locutor de prueba (target) es el LO, siempre y cuando el modelo de ASV esté correctamente entrenado.

3.2.3.2. Histograma de puntuaciones del locutor impostor agregado

El histograma de puntuaciones del LI se calcula teniendo en cuenta que cada LI tiene asociadas 500 sentencias diferentes y que cada sentencia se confronta con respecto a los 20 LOs, es decir, cada LI se hace pasar por los 20 LOs inscritos en el sistema. Por consiguiente, como se presentan 20 LIs se tiene un total de 200000 puntuaciones. Es razonable decir que si el sistema ASV está correctamente entrenado para identificar de forma verídica y unívoca a los múltiples locutores estas puntuaciones deberían estar en torno a 0.

3.2.3.3. Histograma de puntuaciones del locutor objetivo segregado por sexo

El histograma de puntuaciones del locutor objetivo, tanto masculino como femenino, se calcula seleccionando los 10 LOs masculinos o femeninos existentes. Para cada LO seleccionado, se realiza el cálculo de la puntuación generada por el sistema ASV. Dado que cada LO tiene 200 sentencias, se generan 200 puntuaciones para cada LO al confrontar estas sentencias con respecto al *embedding prototipo* del LO. Como resultado, se obtienen un total de 2000 puntuaciones para los LOs seleccionados.

3.2.3.4. Histograma de puntuaciones del locutor impostor segregado por sexo

El histograma de puntuaciones del locutor impostor, tanto masculino como femenino, se calcula seleccionando los 10 LIs masculinos o femeninos existentes. Para cada LI seleccionado, se realiza el cálculo de la puntuación generada por el sistema ASV. Dado que cada LI tiene 500 sentencias, se generan 500 puntuaciones para cada LI al confrontar estas sentencias con respecto al *embedding prototipo* del LO. En este caso el LI masculino o femenino se compara con respecto a los 10 LOs masculinos o femeninos, consecutivamente. Producíendose un total de 100000 puntuaciones.

3.2.3.5. Histograma de puntuaciones conjunto del LO, LI y la EER

Una vez obtenido el histograma de puntuaciones del locutor objetivo agregado y el histograma de puntuaciones del locutor impostor agregado para el sistema de ASV dado, es posible analizar el comportamiento del sistema de ASV cuando un locutor impostor solicita acceso. Para ello, se unifican ambas distribuciones en una misma gráfica, obteniendo así el histograma de puntuaciones conjunto de LOs y LIs, también denominado histograma de puntuaciones del verificador. Esto permite evaluar la robustez y el desempeño del sistema ASV ante intentos de verificación de locutores no registrados.

Cabe destacar que el LI no intenta imitar al LO, es decir, no se está utilizando ninguna técnica de ‘spoofing’. Estas técnicas de ‘spoofing’ se abordarán más adelante. Por lo tanto, este análisis constituye una evaluación de referencia del sistema de ASV independiente del sexo del locutor.

Un aspecto muy importante a tener en cuenta es que para la generación del histograma del LO se utilizan 4000 puntuaciones, mientras que para la generación del histograma del LI se utilizan 200.000 puntuaciones. Por consiguiente, como el número de puntuaciones obtenidas en el LO es mucho menor al número de puntuaciones del LI, se interpreta cada histograma como una aproximación de distribución de probabilidad, esto permite comparar distribuciones de diferentes tamaños directamente. Si se tiene un número de puntuaciones z . En este histograma si en un bin se tiene una frecuencia x , esto indica que existen x puntuaciones en dicho bin. Al convertir la frecuencia en una densidad de probabilidad, si el ancho del bin es y , la densidad para ese bin es $x/(z * y)$. Esta aproximación también se aplica en la obtención del histograma de puntuaciones conjunto de LOs,LIs segregado por sexo, ya que en dicho caso son 2000 puntuaciones frente a 100.000 puntuaciones.

Para la obtención de la EER se realiza la gráfica de tasa de falsos positivos (FPR) y tasa de falsos negativos (FNR) en función del umbral de puntuación. Para ello, se utilizan las puntuaciones generadas para el LO y LI. A partir de las puntuaciones del LO se obtiene la tasa de falsos negativos en función del umbral establecido y a partir de las puntuaciones del LI se obtiene la tasa de falsos positivos en función del umbral. Al generar dicha gráfica se observa un punto de intersección entre ambas curvas, este punto define la EER, cuya inversa define la precisión del sistema de ASV.

3.2.3.6. Histograma de puntuaciones conjunto del LO, LI segregado por sexo y la EER

Una vez obtenido el histograma de puntuaciones del locutor objetivo (masculino o femenino) y el histograma de puntuaciones del locutor impostor (masculino o femenino) para el sistema de ASV dado, es posible analizar el comportamiento del sistema de ASV cuando un locutor impostor, ya sea masculino o femenino, solicita acceso. Para ello, se unifican ambas distribuciones en una misma gráfica, obteniendo así el histograma de puntuaciones conjunto de LOs y LIs segregado por sexo (masculino o femenino). Esto permite evaluar la robustez y el desempeño del sistema de ASV ante intentos de verificación de locutores no registrados, considerando la variable del sexo. Este análisis constituye una evaluación de referencia del sistema de ASV dependiente del sexo del locutor.

Para la obtención de la EER se realiza la gráfica de tasa de falsos positivos (FPR) y tasa de falsos negativos (FNR) en función del umbral de puntuación. Para ello, se utilizan las puntuaciones generadas para el LO y LI masculino o femenino. A partir de las puntuaciones del LO se obtiene la tasa de falsos negativos en función del umbral establecido y a partir de las puntuaciones del LI se obtiene la tasa de falsos positivos en función del umbral. Al generar dicha gráfica se observa un punto de intersección entre ambas curvas, este punto define la EER.

3.2.4. Resultados

En la Figura 3.1 se muestran los histogramas de puntuaciones conjunto de LO, LI, segregado por sexo masculino, para diferentes modelos ASV. Los resultados parten de analizar 10 locutores objetivos masculinos frente a 10 locutores impostores masculinos. En la Figura 3.2 se muestra la FPR, FNR en función del umbral, segregado por sexo masculino, para diferentes modelos ASV. Además, se representa la EER y el umbral de puntuación asociado.

En la Figura 3.3 se muestran los histogramas de puntuaciones conjunto de LO, LI, segregado por sexo femenino, para diferentes modelos ASV. Los resultados parten de analizar 10 locutores objetivos femeninos frente a 10 locutores impostores femeninos. En la Figura 3.4 se muestra la FPR, FNR en función del umbral, segregado por sexo femenino, para diferentes modelos ASV. Además, se representa la EER y el umbral de puntuación asociado.

En los histogramas conjuntos de LO, LI, segregados por sexo (Figuras 3.3, 3.1), se observa como la puntuación media del LO y LI es aproximadamente equivalente, tanto en el caso masculino como en el femenino, independientemente del sistema ASV utilizado, además, la desviación estándar de las distribuciones en ambos sexos es semejante, todo ello implica que el cruce de los histogramas de puntuaciones del LO y LI sea equivalente. Presentándose así la misma precisión del sistema de ASV independientemente del sexo utilizado en el análisis. Este fenómeno se corrobora representando el FPR, FNR de las puntuaciones del LI y LO, respectivamente. (Figuras 3.4, 3.2). Se observa como la EER es equivalente en ambos sexos, en los diferentes modelos ASV analizados.

El histograma del LO conlleva una baja desviación respecto al valor medio de la puntuación asociada al LO cuando se utiliza el sistema ASV x-vector (Figura 3.5a), esto plasma una buena identificación del LO, la cual es mejor que en los demás

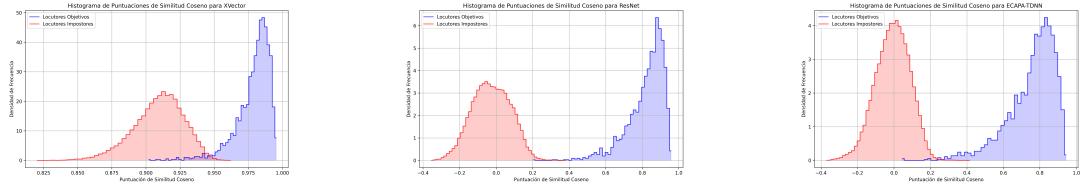


Figura 3.1: Histogramas de puntuaciones LI, LO (segregado por sexo: Masculino) para diferentes modelos ASV

sistemas ASV (ResNet y ECAPA-TDNN (Figuras 3.5b, 3.5c), pero el LI implica una alta puntuación, cercana a la del LO, cosa que no sucede en ResNet y ECAPA-TDNN, cuyo valor medio de puntuación es aproximadamente nulo. En conclusión, aunque la desviación estándar del histograma de puntuación del LO en el sistema x-vector es menor que en los sistemas ResNet y ECAPA-TDNN, la puntuación media del LI es significativamente mayor en el sistema x-vector. Esto resulta en que la precisión de verificación del sistema x-vector sea inferior a la de los sistemas ResNet y ECAPA-TDNN.

En el caso de que se desee que el sistema x-vector no permita la entrada de ningún LI, se puede aumentar el umbral de puntuación. Por ejemplo, si se observa la Figura 3.5a , a partir de un valor de puntuación de 0,95, únicamente hay distribución por parte del LO, mientras que el LI no presenta ninguna puntuación a partir de este valor. Por tanto, si se establece este valor de puntuación como umbral de decisión, nunca se contabilizará un falso positivo, pero sí aumentará la tasa de falsos negativos. Esto, a su vez, incrementará la EER, disminuyendo así la precisión del sistema x-vector. Este comportamiento se debe a que la tasa de falsos negativos aumenta con la disminución de la tasa de falsos positivos y viceversa.

Para poder hablar de la robustez práctica del sistema ASV es necesario realizar ataques de ‘spoofing’ al sistema ASV, de tal manera que un locutor target (LT) se haga pasar por un locutor objetivo, siendo este un locutor impostor. De esta manera, es posible analizar cómo se ve disminuida la robustez de los sistemas ASV de referencia. Se ha analizado como la EER es prácticamente nula en el sistema ASV ResNet y ECAPA-TDNN y en el sistema de ASV x-vector, se presenta una EER del 2,43 %, aunque sigue siendo un valor muy bajo, pero estas medidas de EER obtenidas para los tres sistemas ASV analizados suceden cuando el LI no se hace pasar por el locutor objetivo, por consiguiente, es posible decir que los tres sistemas ASV estudiados son robustos cuando no existe suplantación de identidad, en otras palabras, están correctamente entrenados dichos sistemas para identificar locutores, pero es necesario estudiar la robustez del sistema de verificación de locutor automática cuando se trata de realizar suplantación de identidad, para ello, se van a utilizar dos sistemas de TTS. El primer sistema está formado por FastSpeech 2 + DiffWave, y el segundo sistema está compuesto por Tacotron 2 + HiFi-GAN.

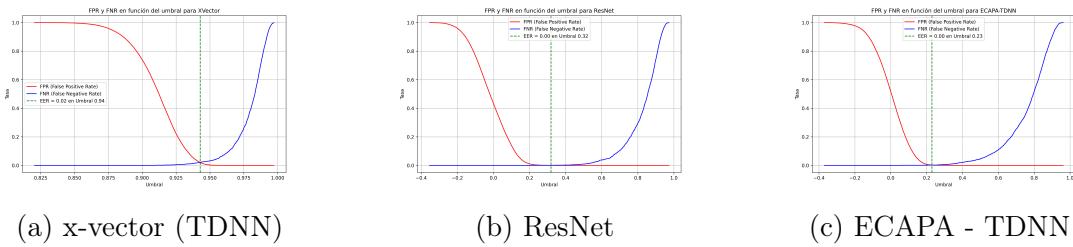


Figura 3.2: Tasa de falsos positivos (FPR) y Tasa de falsos negativos (FNR) en función del umbral para diferentes modelos ASV (segregado por sexo: Masculino)

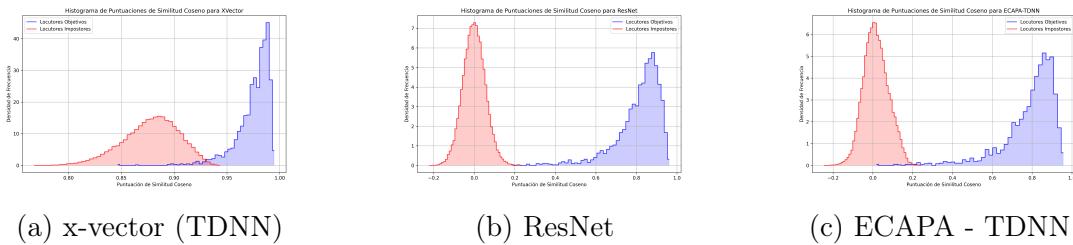


Figura 3.3: Histogramas de puntuaciones LI vs LO (segregado por sexo: Femenino) para diferentes modelos ASV

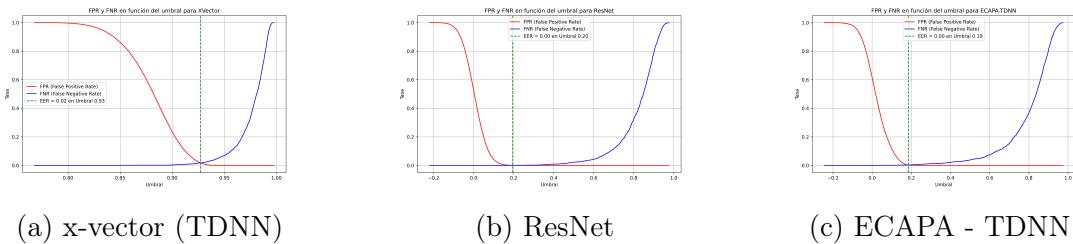


Figura 3.4: Tasa de falsos positivos (FPR) y Tasa de falsos negativos (FNR) en función del umbral para diferentes modelos ASV (segregado por sexo: Femenino)

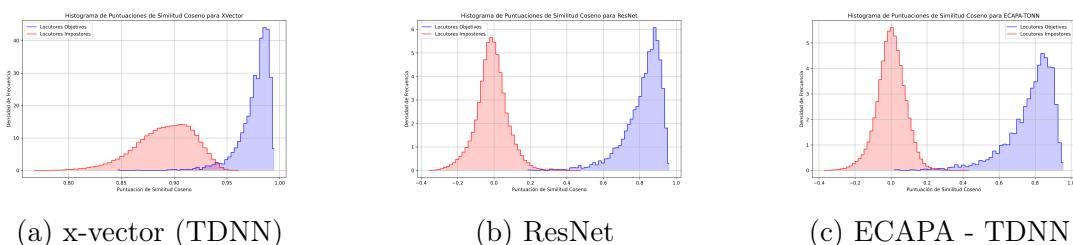


Figura 3.5: Histogramas de puntuaciones LI, LO (Gráfica agregada) para diferentes modelos ASV

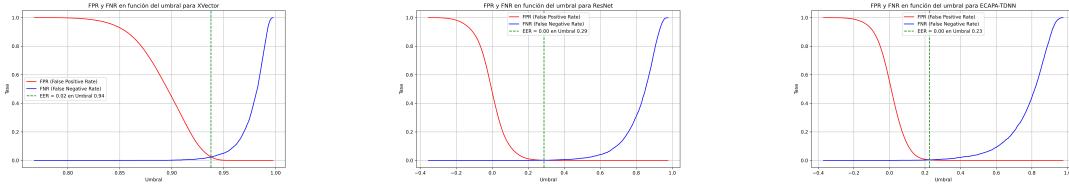


Figura 3.6: Tasa de falsos positivos (FPR) y Tasa de falsos negativos (FNR) en función del umbral para diferentes modelos ASV (Gráfica agregada)

3.2.4.1. Comparación de la EER obtenida con la EER proporcionada por el repositorio

En este apartado se realiza la comparación de la tasa de error equivalente (EER) calculada en los diferentes sistemas de ASV utilizados: x-vector, ResNet y ECAPA-TDNN, con respecto a la tasa de error equivalente (EER) calculada por los autores de dichos sistemas pre-entrenados.

En los diferentes repositorios utilizados, en los cuales se encuentran los sistemas de ASV analizados pre-entrenados [44],[43], se presentan las tasas de error equivalente (EERs) obtenidas en la fase de test de dichos modelos. Los tres modelos de ASV estudiados se han entrenado mediante la base de datos VoxCeleb (Voxceleb1+Voxceleb2), pero utilizando diferentes tipos de redes codificadoras (TDNN, ResNet y ECAPA-TDNN). Todos estos sistemas de verificación pre-entrenados surgen de *Speechbrain*³, donde el sistema entrena una red neuronal para la generación de *embeddings* de locutores junto con un clasificador de identificación de locutor.

Todos los resultados que se presentan en la Tabla 3.3 se obtuvieron con la división de verificación oficial de voxceleb1 (*veri_test2.txt*). Dicho fichero conlleva la siguiente estructura: <etiqueta><ruta_archivo1><ruta_archivo2>

- <etiqueta> es un número entero que puede ser 0 o 1. Si vale 1 significa que es un locutor inscrito, si es 0, significa que es un locutor impostor.
- <ruta_archivo1> es una cadena de texto que representa la ruta del primer archivo de audio.
- <ruta_archivo2> es una cadena de texto que representa la ruta del segundo archivo de audio.

La EER se calcula de la siguiente manera en el repositorio: cada LO se evalúa un número determinado de veces y , siendo y variable, y con valor alto. En este fichero se evalúan 40 LOs y se presentan un total de 37611 identificaciones. Un número determinado de veces x , se comparan sentencias del LO con otra sentencia diferente del mismo. Siendo x la mitad de y , puesto que las verificaciones de locutor se producen a pares, es decir, primero una comparación del LO con respecto al LO, y después una comparación del LO con respecto al LI.

En el primer caso (comparación LO-LO), la etiqueta asociada es 1, esto quiere decir que al generar ambas sentencias, crear los correspondientes embeddings, y

³Speechbrain es un kit de herramientas de código abierto para el desarrollo y la investigación en el campo del procesamiento del lenguaje natural y el reconocimiento de voz. Desarrollado en PyTorch, Speechbrain proporciona una plataforma unificada para construir, entrenar y desplegar modelos de reconocimiento de voz, síntesis de voz, y otras aplicaciones relacionadas con el audio y el habla.

compararlos mediante la distancia coseno, debe dar una puntuación tal que supere el umbral pre-establecido, para que la predicción del ASV sea 1, es decir, ambos locutores, el target y el inscrito son equivalentes. Por tanto, en dicha situación no se presenta ningún error en la decisión del verificador (un positivo verdadero más). En el caso de que la predicción sea 0, como la etiqueta es 1, se contabiliza un error (un falso negativo más). La tasa de falsos negativos (FNR) se calcula a partir de la Ecuación 2.4 , donde se contabiliza el número de falsos positivos y se divide con respecto al número total de accesos x , por parte de dicho LO para un umbral determinado. Al tomar como variable el umbral de decisión, se obtiene un vector de valores de FNR en función del umbral. De esta manera se analiza el comportamiento general del sistema verificador cuando el locutor de prueba es un locutor inscrito en el sistema de ASV.

En el segundo caso (comparación LO-LI), la etiqueta asociada es 0, y por tanto, si el sistema de ASV genera como predicción un 0, se contabiliza un verdadero negativo (TN), mientras que si la predicción es 1, se contabiliza un falso positivo (FP). Al realizar x comparaciones se obtiene la tasa de falsos positivos para un umbral determinado, según la Ecuación 2.5. Este mismo procedimiento se repite para diferentes valores de umbral de puntuación, para así establecer el comportamiento general del sistema verificador cuando el locutor de prueba es un locutor impostor.

Cada sentencia del LO se utiliza 8 veces. Por consiguiente, se utilizan para cada LO (inscrito) $y/8 = z$ sentencias/locutor objetivo. La misma sentencia del LO se confronta 4 veces con diferentes sentencias del mismo LO y se confronta 4 veces más con diferentes sentencias de diferentes LIs.

Sistema	Base de Datos	EER [%]
Xvector + PLDA	VoxCeleb 1	3,23
ECAPA-TDNN	VoxCeleb 1	0,80
ResNet TDNN	VoxCeleb 1	0,95

Tabla 3.3: EER obtenida en los sistemas ASV estudiados,(veri_test2.txt)

Sistema	Base de Datos	EER [%]
Xvector + PLDA	VoxCeleb 2, LibriTTS	2,43
ECAPA-TDNN	VoxCeleb 2, LibriTTS	0,27
ResNet TDNN	VoxCeleb 2, LibriTTS	0,06

Tabla 3.4: EER obtenida en los sistemas ASV estudiados,(Resultados propios)

Al comparar las Tablas 3.3 y 3.4 , se puede decir que obtienen aproximadamente las mismas EERs utilizando la base de datos de VoxCeleb 1 que utilizando las bases de datos VoxCeleb 2 y LibriTTS, aunque la medición de la EER calculada experimentalmente es un poco menor que la EER estimada por parte de los creadores de dichos modelos ASV.

Los sistemas de ASV analizados se entrenaron con VoxCeleb1 (VoxCeleb1-{E,H}) y VoxCeleb2. La verificación del sistema de ASV entrenado se realiza con la base de datos VoxCeleb1-O, la cual está asociada al fichero de división de verificación

oficial de VoxCeleb1 (veri_test2_.txt). Con esto se quiere decir que la evaluación de la precisión del sistema se realiza con datos que no provienen del conjunto de entrenamiento, implicando así un análisis más riguroso. En cambio, la fase de evaluación propuesta en este trabajo se basa en que los locutores impostores parten de la base de datos VoxCeleb2 y por tanto, es sabido, que estos locutores los va a identificar con mayor precisión (menor tasa de errores), porque ha aprendido en la fase de entrenamiento a codificar y caracterizar estos locutores. Pero, por otra parte se ha realizado el proceso de inscripción de nuevos locutores, nunca aprendidos por el sistema de ASV, puesto que provienen de la base de datos LibriTTS.

Como conclusión, es probable que el experimento propuesto plantea una menor EER que respecto al propuesto por el repositorio puesto que en el repositorio se testea con datos no aprendidos, mientras que el experimento propuesto evalúa locutores aprendidos (VoxCeleb2) con locutores no aprendidos (LibriTTS).

Además, es muy importante cómo se evalúa la EER. La EER estimada en el repositorio parte del análisis de 40 locutores provenientes de VoxCeleb1-O. Se generan 37611 puntuaciones, aproximadamente unas 18800 puntuaciones para la tasa falso rechazo y otras 18800 para la tasa de falsa aceptación. Con estas puntuaciones se establece la EER.

En cambio, en el caso de estudio se utilizan 20 LOs (10 masculinos y 10 femeninos) provenientes de la base de datos LibriTTS, donde cada uno de los locutores objetivos se inscribe generando un *embedding prototipo*, a partir de un determinado número de sentencias. Cada uno de los LOs utiliza 200 sentencias para evaluar sus puntuaciones, obteniendo un total de 4000 puntuaciones para el cálculo de la tasa de falso rechazo. Por otra parte, se presentan 20 LIs que parten de la base de datos VoxCeleb2 y se evalúan 500 sentencias de cada LI con respecto a los 20 *embeddings* prototipos de LO, generando un total de 200.000 puntuaciones.

Como conclusión, decir que la evaluación de la precisión del sistema de ASV realizada es diferente y es posible decir que el experimento realizado es más formal, debido a que se realiza la inscripción de un nuevo locutor. En el repositorio no se emula dicha situación y se generan los *embeddings prototipos* de los LOs a partir de una única sentencia.

3.3. Estudio del rendimiento de los sistemas de ASV utilizados frente a ataques de spoofing

En este apartado se realizará un estudio de la precisión de los sistemas de verificación automática de locutor (ASV) frente a ataques de spoofing, utilizados en la fase experimental, basándose en el marco general utilizado en estudios de ataques de suplantación, Sección 2.2.4, para ello se analizará el rendimiento del sistema de ASV en condiciones de suplantación.

Inicialmente, se realizará la síntesis de voz del locutor proveniente de base de datos LJSpeech. LJSpeech está compuesta por 24 horas de una locutora leyendo. Se utilizarán modelos pre-entrenados de dicha base de datos en el sistema de síntesis de voz FastSpeech 2, y en el vocoder Diffwave. El sintetizador de voz se va a utilizar para generar la predicción del espectrograma a partir del modelo pre-entrenado y un texto asociado. Utilizando el vocoder se obtiene la muestra de audio sintetizada

a partir del modelo pre-entrenado y la predicción del espectrograma generada por FastSpeech 2. Una vez generada la voz sintética, se utiliza para ejecutar ataques de spoofing en los diferentes sistemas de ASV estudiados.

Después, se va a realizar la síntesis de voz de distintos locutores provenientes de la base de datos LibriTTS, concretamente, 20 locutores (10 masculinos y 10 femeninos) para poder analizar la calidad de la síntesis de voz 'multi-speaker' y el rendimiento del modelo de ASV en función del sexo del locutor suplantado. El sistema de TTS utilizado es Tacotron 2 + HiFi-GAN.

3.3.1. Síntesis de voz de la base de datos LJSpeech utilizando FastSpeech2 y DiffWave

En este apartado se describen primeramente los métodos utilizados para generar los resultados experimentales, después se explican los resultados derivados de los diferentes métodos utilizados, con el fin de evaluar el rendimiento de precisión de los diferentes sistemas de ASV frente a ataques de suplantación de identidad. Para ello se realiza la síntesis de voz del locutor de la base de datos LJSpeech utilizando FastSpeech 2 y DiffWave.

3.3.1.1. Métodos

Síntesis de voz: La síntesis de voz de la locutora existente en la base de datos LJSpeech se realiza utilizando los modelos pre-entrenados y las herramientas necesarias. Para generar texto variable y con una longitud determinada se utiliza el *Corpus Brown*⁴. Una vez generadas las 1000 sentencias diferentes en texto y precargados los modelos de LJSpeech en FastSpeech 2 y DiffWave, se utiliza cada una de las sentencias generadas para realizar la inferencia de codificación del texto con FastSpeech 2, aplicando los valores de los parámetros 'pace', 'pitch_rate' y 'energy_rate'. Como resultado, se obtiene una predicción del espectrograma de Mel, junto con otras salidas, como estimaciones de duración, tono y energía. 'pace' ajusta la velocidad de la voz generada, por defecto es 1, que significa velocidad normal, 'pitch_rate' modifica la altura del tono, por defecto es 1, que significa tono normal y 'energy_rate' ajusta la energía de la voz, cuyo valor es 1 para una energía normal.

Una vez obtenida la predicción del espectrograma de Mel, esta se utiliza en el vocoder DiffWave en modo de decodificación por lote (batch), es decir, para convertir el espectrograma en una forma de onda. Además, para que se pueda realizar la decodificación correctamente es necesario definir la longitud de superposición, de acuerdo con el utilizado al extraer el espectrograma Mel. Adicionalmente, existe un parámetro de decisión, para realizar un muestreo más rápido y eficiente (modo de muestreo rápido). En esta situación hay que definir una lista de valores que controla el nivel de ruido durante el modo de muestreo rápido, afectando esto a la calidad del audio generado.

⁴El Corpus Brown es una colección de aproximadamente 1 millón de palabras de inglés americano, creada en 1961 por Henry Kučera y Nelson Francis. Diseñado para representar diversos géneros textuales, sirve como recurso fundamental en lingüística y procesamiento del lenguaje natural, desarrollo de modelos de lenguaje e investigación empírica

Histograma de puntuaciones del locutor objetivo: La generación del histograma de puntuaciones del LO se calcula inicialmente realizando la inscripción de la base de datos LJSpeech en los diferentes sistemas ASV estudiados (x-vector, ResNet y ECAPA-TDNN), para ello, se genera el *embedding prototipo* utilizando 1000 sentencias diferentes de dicha locutora y realizando la media estadística de los 1000 *embeddings* resultantes. Después, se comparan 1000 *embeddings* generados del LO con respecto al *embedding prototipo* del LO, obteniéndose así 1000 puntuaciones cuando se realiza la distancia coseno entre ambos *embeddings*. De esta manera se puede obtener el histograma de puntuaciones para el LO.

Histograma de puntuaciones del locutor objetivo sintetizado: La creación del histograma de puntuaciones del LOS se obtiene a partir de las 1000 sentencias generadas sintéticamente del locutor objetivo. Estas 1000 sentencias generan 1000 *embeddings* según el sistema ASV utilizado. Después, estos *embeddings* se comparan con respecto al *embedding prototipo* del sistema ASV asociado realizando la distancia coseno, obteniéndose así 1000 puntuaciones (scores). A partir de estas 1000 puntuaciones se genera el histograma de puntuaciones del LOS.

Histograma de puntuaciones conjunto del LO, LOS y la EER: El histograma de puntuaciones conjunto del LO, LOS, se obtiene unificando en una misma gráfica el histograma de puntuaciones del LO y el histograma de puntuaciones del LOS.

A partir de las puntuaciones del LO y del LOS se puede obtener la gráfica de la tasa de falsos positivos (FPR) y tasa de falsos negativos (FNR) en función del umbral de puntuación. El punto de intersección entre las curvas FPR y FNR define la EER.

Además, se realiza la gráfica DET asociada a los tres sistemas de ASV analizados, para describir el comportamiento operativo de los diferentes sistemas. En dicha gráfica se muestra tanto la EER, como la dependencia de la FNR en función de la FPR. Mediante esta gráfica, es fácil visualizar qué sistema es más preciso, debido a que se presentan diferentes curvas en función del sistema ASV utilizado, y el que menos área bajo la curva (AUC) presente, es el sistema más preciso.

Inscripción del locutor objetivo: En este apartado se define el método realizado para establecer la importancia que conlleva una correcta inscripción por parte de un usuario en el sistema de ASV. Para ello, se realizará una comparación entre la generación del *embedding prototipo* del locutor objetivo utilizando 1000 sentencias y utilizando 100 sentencias, es decir, con 10 veces menos cantidad de información.

La inscripción (o enrollment) se crea calculando un *embedding prototipo* asociado al usuario a inscribir (locutor objetivo o legítimo). En los métodos explicados anteriormente se han utilizado 1000 sentencias diferentes del locutor de estudio, estas 1000 sentencias han generado 1000 *embeddings*, y a partir de estos múltiples *embeddings* se ha calculado el embedding prototipo, utilizando para ello la media estadística. De esta manera, se caracteriza al locutor objetivo independientemente de lo que diga (del contexto), haciendo que se obtenga una identificación fiel del locutor. Por consiguiente, es muy importante realizar una inscripción completa del locutor para que la robustez del sistema ASV no se vea altamente perjudicada.

Se demostrará que al realizar una inscripción menos robusta del locutor objetivo, el sistema ASV se ve más afectado en robustez (precisión de identificación) que respecto a cuando se utiliza una inscripción más compleja. Para ello, se va a utilizar como *embedding prototipo* del locutor objetivo, la media estadística de los 100 *embeddings* asociados a 100 sentencias diferentes del locutor. Este *embedding* prototipo se compara con respecto a 1000 *embeddings* de dicho locutor, generados a partir de 1000 sentencias diferentes, obteniéndose a partir de las puntuaciones resultantes el histograma de puntuaciones del LO.

Por otra parte, se generan 1000 sentencias sintetizadas diferentes de dicho locutor, estas se implementan en el modelo de ASV, para generar 1000 *embeddings*, y estos se comparan con el *embedding prototipo* para así obtener el histograma de puntuaciones del LOS.

3.3.1.2. Resultados

El histograma conjunto de puntuaciones para el LO y LOS evaluado en los diferentes sistemas de ASV se presenta en la Figura 3.7, donde las distribuciones azules hacen referencia al histograma de puntuaciones del LO y las distribuciones rojas hacen referencia al histograma de puntuaciones del LOS. Asociado a cada histograma se ilustra la FPR y FNR en función del umbral.

En la Figura 3.9a se ilustra la familia de curvas DET asociadas a los modelos ASV estudiados cuando se utiliza un *embedding prototipo* creado a partir de 1000 sentencias diferentes, utilizando suplantación de voz mediante síntesis de voz producida por el sistema de TTS FastSpeech 2 + DiffWave. La curva DET establece la relación de la FPR con respecto a la FNR. El punto donde FPR es equivalente a FNR establece la EER, esto sucede teóricamente. En la práctica, la EER se halla para el valor de FNR y FPR tal que la distancia entre ambas tasas sea mínima, para un determinado umbral. En la curva DET experimental y teórica se cumple que la EER es el punto de cruce entre una recta diagonal que parte desde la puntuación (0, 0) y termina en (1, 1) y la curva DET asociada. Como en este experimento se utilizan tres modelos de ASV, se presentan tres curvas DET, y por tanto se producen tres puntos de cruce de la recta diagonal con respecto a la familia de curvas DET.

Para poder cuantificar cuánto se ha visto perturbada la robustez de los sistemas ASV al realizar el ataque de suplantación de identidad, se va a comparar la EER obtenida cuando se analizaban locutores objetivos respecto a locutores impostores, los cuales no realizaban ningún ataque de suplantación de identidad de locutores inscritos en el ASV (spoofing), con respecto a cuando se realiza suplantación de identidad (síntesis de voz). Es decir, comparación del rendimiento del sistema de referencia o estándar con respecto al rendimiento del sistema en condición de suplantación.

Es posible observar en la Tabla 3.5 cómo se ha visto disminuida la robustez de los sistemas ASV al utilizar como técnica de spoofing la síntesis de voz de un locutor objetivo, el cuál estaba inscrito en el sistema.

En parte, el sistema de ASV no se ha visto altamente perjudicado puesto que la clonación podría ser mejor, teniendo en cuenta los valores obtenidos de pérdida de validación (valid), donde se obtiene un valor de pérdida de espectrograma Mel postNet de 0.950. Una pérdida de alrededor de 0.95 podría indicar que hay espacio para mejorar la calidad del espectrograma Mel generado por el modelo. Este error

	EER [%]	EER Suplantación [%]
xVector	2,43	3,90
ResNet	0,27	9,30
ECAPA	0,06	12,40

Tabla 3.5: Comparación de la EER estimada sin técnicas de Spoofing vs. con suplantación de identidad

incurre en un error a la hora de generar la forma de onda a partir del espectrograma Mel estimado, es por ello que la calidad del audio sintetizado no es demasiado real. Debido a esto, el sistema de ASV es capaz de discernir con una alta probabilidad entre locutor objetivo y locutor impostor, aunque el locutor impostor trate de suplantar la identidad del locutor objetivo de forma veraz. El sistema de ASV más perjudicado es el modelo ECAPA-TDNN, con una EER del 12,40 %, esto quiere decir que la precisión del sistema de ASV basado en ECAPA-TDNN es del 87,6 %. Análogamente, el sistema de ASV basado en ResNet tiene una precisión del 90,7 % y el sistema de ASV basado en x-vector tiene una precisión del 96,1 %.

En la Figura 3.9 se ilustra una comparación de las curvas DET obtenidas para dos tipos diferentes de inscripciones del locutor objetivo, utilizando ataques de suplantación en el sistema de ASV. Donde, la Figura 3.9a muestra las curvas DET correspondientes a los tres sistemas de ASV analizados, utilizando una inscripción del locutor objetivo con 1000 sentencias diferentes y la Figura 3.9b muestra las curvas DET correspondientes a los tres sistemas de ASV analizados, utilizando una inscripción del locutor objetivo con 100 sentencias diferentes. Es posible observar a simple vista como el sistema de ASV con una inscripción del locutor de 1000 sentencias tiene mayor precisión que el sistema de ASV con una inscripción que utiliza 10 veces menos de sentencias. Teóricamente, el sistema de mayor precisión es el que tiene menor área bajo la curva (AUC) DET. En el caso de que se graficara la curva ROC, el mejor sistema sería el que mayor AUC implique. Además, se observa que el sistema de ASV menos afectado es el que utiliza la arquitectura codificadora TDNN.

En la Figura 3.8 se ilustran los histogramas conjuntos de puntuaciones del LO y LOS, para diferentes sistemas de ASV, cuando la inscripción del LO se genera utilizando 100 sentencias diferentes. Además, se ilustra la gráfica de FPR y FNR en función del umbral de puntuación para cada uno de los sistemas de ASV utilizados.

A modo de comparación, se muestra la Tabla 3.5, donde se muestra la EER obtenida cuando se utilizan 1000 sentencias para inscribir al locutor objetivo y cuando se utilizan 100 sentencias para inscribirlo.

	EER [%] EP-1000 sentencias	EER [%] EP-100 sentencias
x-vector	3,90	3,90
ResNet	9,30	15,10
ECAPA-TDNN	12,40	18,30

Tabla 3.6: Comparación de la EER estimado con embedding Prototipo (EP) compuesto por 1000 Sentencias vs. 100 Sentencias

Se puede deducir a partir de la Tabla 3.6 que al disminuir el número de sentencias asociadas al cálculo del *embedding prototipo* del locutor inscrito, la EER aumenta sustancialmente en los sistemas ResNet y ECAPA-TDNN (un aumento de la EER del 5,80 % y 5,90 %, respectivamente), esto se traduce en una disminución de la robustez y el rendimiento del sistema de ASV.

Por otra parte, debe notarse que la robustez en el sistema x-vector es invariante frente a la variación de cantidad de sentencias para calcular el *embedding prototipo*, por consiguiente, se puede decir como conclusión inicial, que el sistema x-vector es capaz de definir de manera eficaz la identidad del locutor, puesto que no son necesarias tanta cantidad de sentencias asociadas al locutor objetivo para poder caracterizar correctamente al locutor objetivo. Sin embargo, en los sistemas ResNet y ECAPA-TDNN si son necesarias una mayor cantidad de sentencias para poder inscribir verazmente al locutor objetivo, para que sea más difícil suplantar la identidad de dicho locutor objetivo, por tanto, los sistemas ResNet y ECAPA-TDNN son menos eficientes en términos de inscripción de un locutor que con respecto al sistema de ASV basado en x-vector.

En conclusión, el sistema de ASV que mejor se comporta ante ataques de suplantación de identidad cuando se realiza la síntesis de voz de un locutor inscrito es x-xector, con una EER del 3,90 % , después ResNet con una EER estimado del 9,30 % y finalmente, el sistema menos preciso es ECAPA-TDNN con una EER del 12,40 %.

Hay que tener en cuenta que este experimento únicamente se ha realizado con un locutor, es por ello que presenta peso empírico limitado. Sería necesario realizar un análisis más amplio, en el cual se incluyan múltiples locutores inscritos y sus respectivas sintetizaciones, para evaluar correctamente la robustez y rendimiento de los diferentes sistemas de ASV frente a ataques de suplantación. Esta es la razón por la cual se presenta un experimento asociado a este análisis.

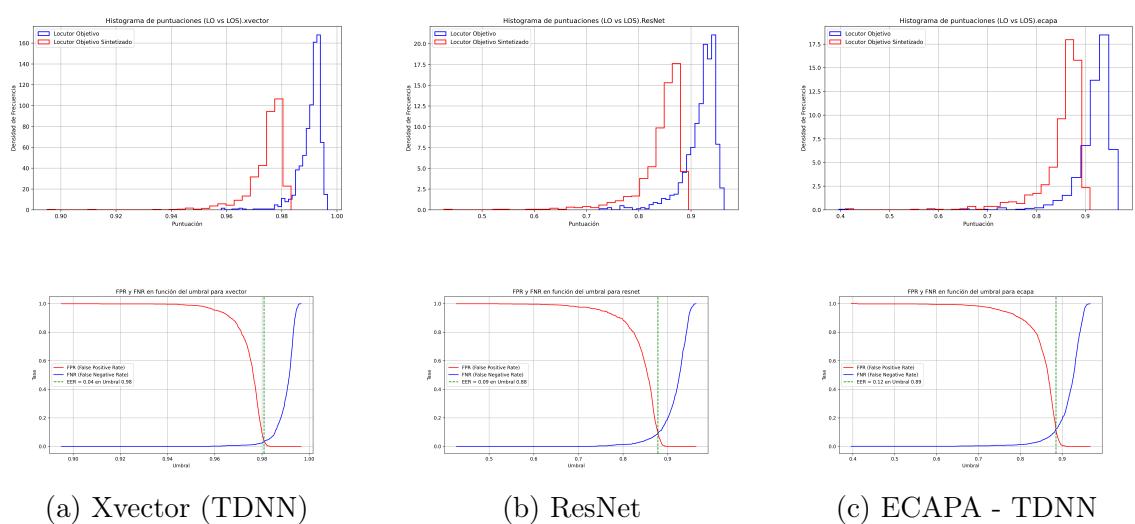


Figura 3.7: Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs (umbral). EP-1000 sentencias

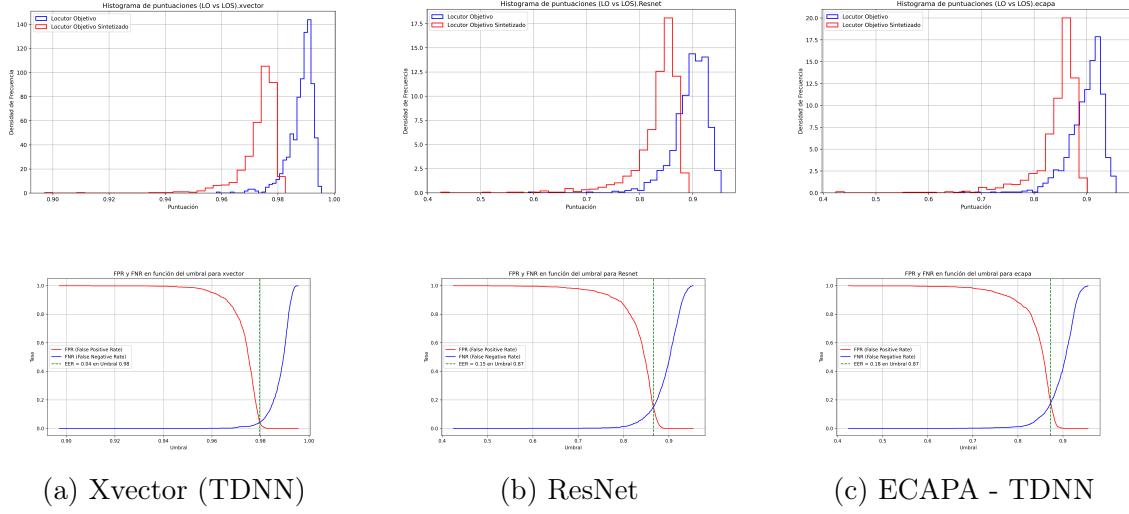


Figura 3.8: Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs (umbral). EP-100 sentencias

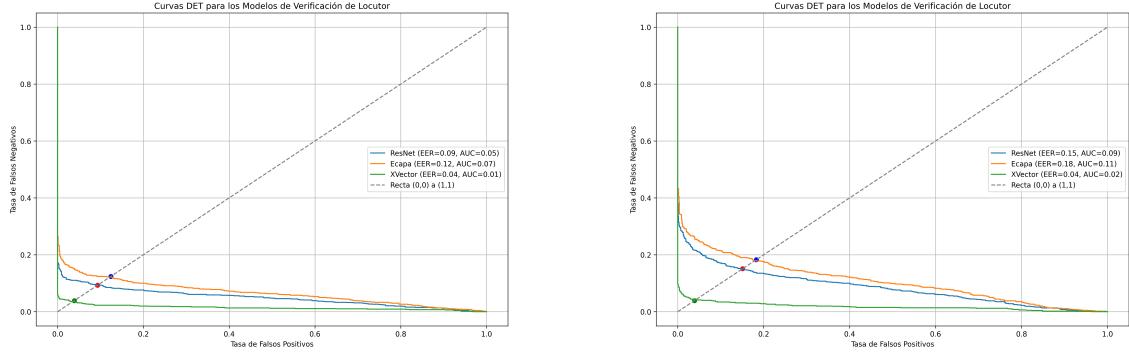


Figura 3.9: Comparación de las curvas DET para diferentes inscripciones del LO utilizando síntesis del LO mediante FastSpeech2 y Diffwave.

3.3.2. Síntesis de voz de la base de datos LibriTTS utilizando Tacotron2 y HiFiGAN

Para realizar este experimento, primeramente se realizará la síntesis de voz de los 20 locutores objetivo (inscritos en el sistema ASV, la inscripción se realizó en la evaluación del rendimiento del sistema de ASV). El análisis de la robustez y rendimiento del sistema se separará en 2 partes, segregado por sexo (femenino y masculino, por separado) y agregado (femenino y masculino, conjuntamente). Es decir, un experimento dependiente del sexo del locutor y otro independiente del sexo del locutor, para analizar la respuesta del sistema de ASV en función del sexo y observar posibles diferencias que pudieran aparecer respecto a la precisión del sistema de ASV. Para ello, se utilizarán las típicas gráficas de FPR y FNR en función del umbral de puntuación, para obtener la EER, siendo esta una medida objetiva del rendimiento de la precisión del sistema de verificación del locutor analizado.

3.3.2.1. Métodos

Síntesis de voz: Cada LO se sintetizará con 1000 sentencias diferentes. Estas sentencias se generan a partir del corpus Brown. Para obtener la síntesis de voz mediante el TTS Tacotron 2, se carga el modelo entrenado con la base de datos LibriTTS. En este modelo, se utiliza un audio de referencia del locutor objetivo a sintetizar (se escoge el audio más largo asociado a la base de datos de cada LO) y la sentencia redactada obtenida del corpus Brown. Como resultado, se obtiene una representación intermedia de la forma de onda sintetizada, específicamente una estimación del espectrograma de Mel. Este espectrograma de Mel se utiliza en el vocoder HiFi-GAN (previamente entrenado en la base de datos LibriTTS) para generar el audio sintetizado, que está asociado con la sentencia y el locutor de referencia implementado.

Histograma de puntuaciones del locutor objetivo sintetizado: Una vez obtenidas las 1000 sentencias sintetizadas de cada uno de los 20 locutores objetivos, se procede a generar los *embeddings* asociados a dichas sentencias. Estas sentencias se utilizan en el modelo pre-entrenado del ASV para obtener los *embeddings* del locutor objetivo sintetizado (LOS). Cada *embedding* generado se compara con respecto al *embedding prototipo* de dicho locutor objetivo utilizando como medida de similitud la distancia coseno. Las puntuaciones resultantes generan el histograma de puntuación para el LOS. Al unificar las puntuaciones de todos los LOSs se obtiene el histograma de puntuaciones para el LOS agregada. Si se unifican únicamente las puntuaciones de los LOSs masculinos, se obtiene el histograma de puntuaciones para el LOS segregado por sexo masculino y si se unifican las puntuaciones de los LOSs femeninos, se obtiene el histograma de puntuaciones para el LOS segregado por sexo femenino.

Histograma de puntuaciones del locutor objetivo: En secciones anteriores, en la Sección de caracterización del locutor objetivo (3.2.1) se han calculado los *embeddings prototipos* de los 20 locutores objetivos del corpus LibriTTS, utilizando para la definición de la inscripción del LO, 100 sentencias y como distribución de puntuaciones del LO, 200 sentencias. Al unificar todas las puntuaciones obtenidas para los LOs se obtiene el histograma de puntuación del LO agregada (Figura 3.5. Gráficas azules). Si se unifican únicamente las puntuaciones de los LOs masculinos, se obtiene la distribución de puntuación para el LO segregado por sexo masculino (Figura 3.1. Gráficas azules). Finalmente, si se unifican las puntuaciones de los LOs femeninos, se obtiene la distribución de puntuación para el LO segregado por sexo femenino (Figura 3.3. Gráficas azules).

Una vez organizadas y dispuestas todas las puntuaciones se puede graficar el histograma conjunto del (LO, LOS) agregado, segregado por sexo, masculino y femenino, y se puede obtener la EER asociada a cada uno de los sistemas de ASV estudiados.

Histograma de puntuaciones conjunto del LOS, LO segregado por sexo masculino: Para graficar la distribución de puntuaciones del LO y LOS, segregado por sexo masculino, se unifican por una parte las puntuaciones obtenidas para los locutores objetivos sintetizados masculinos y así obtener la distribución gene-

ral de puntuaciones del LOS masculino y por otra parte, se utiliza la distribución de puntuaciones del LO, referente a sexo masculino, calculada en 3.2.1, Figura 3.1 (Histogramas azules). Ambas distribuciones (LO y LOS) se plasman en una misma gráfica para observar el comportamiento del sistema de ASV frente a ataques de suplantación de identidad, de sexo masculino. El resultado se presenta en la Figura 3.10.

Histograma de puntuaciones conjunto del LOS, LO segregado por sexo femenino: Para graficar el histograma conjunto (LO, LOS) de puntuaciones segregada por sexo femenino, se realiza la concatenación de todas las puntuaciones asociadas a los locutores objetivos sintetizados femeninos, obteniéndose así el histograma de puntuaciones del locutor objetivo sintetizado femenino. Por otra parte, se reutiliza el histograma de puntuaciones del LO femenino, y se grafican ambas distribuciones, mediante un histograma conjunto para así poder analizar el comportamiento del sistema de ASV frente ataques de suplantación de identidad, de sexo femenino. El resultado se presenta en la Figura 3.11.

Histograma de puntuaciones conjunto del LOS, LO agregado: La obtención del histograma conjunto de puntuaciones de todos los locutores objetivos (LOs) y locutores objetivos sintetizados (LOSs), es decir, el histograma conjunto de locutor objetivo y locutor objetivo sintetizado, agregado, se basa en realizar la unificación de las puntuaciones obtenidas para los locutores objetivos sintetizados femeninos y masculinos, obteniéndose así el histograma de puntuaciones del LOS agregado. Por otra parte, se reutilizan las puntuaciones de todos los locutores objetivos, anteriormente calculados para el estudio del rendimiento de referencia del sistema ASV, obteniéndose así el histograma de puntuaciones del locutor objetivo frente a locutor objetivo sintetizado agregado (locutores masculinos y femeninos). El resultado se presenta en la Figura 3.12.

3.3.2.2. Resultados

En la Tabla 3.7 se representa una comparación de la EER obtenida en los sistemas de ASV utilizados, cuando se realizan sucesivos ataques de spoofing (ataque de spoofing proveniente de locutor de sexo masculino (SM), sexo femenino (SF) y ataque de spoofing proveniente de cualquier locutor), utilizando para ello un sintetizador de voz multi-speaker, basado en Tacotron 2 y HiFi-GAN, pre-entrenados en la base de datos LibriTTS.

	EER [%] (SM)	EER [%] (SF)	EER [%]
TDNN	15,45	15,08	15,47
ResNet	10,63	9,62	10,20
ECAPA	23,11	10,79	16,63

Tabla 3.7: Comparación de la EER estimada, para los diferentes sistemas de ASV, en función del sexo

Se deduce de la Tabla 3.7 que el sistema de ASV que más precisión pierde cuando se realizan ataques de spoofing es, en general, ECAPA-TDNN. Cuando el sexo del

locutor objetivo es masculino el peor sistema de ASV sigue siendo ECAPA-TDNN. Sin embargo, cuando el sexo del locutor objetivo es femenino, la precisión en ResNet y ECAPA-TDNN es muy similar, aunque sigue siendo el peor sistema ECAPA-TDNN. El mejor sistema de ASV, en rendimiento bajo suplantación, es ResNet con una EER del 10,20 %.

En la Figura 3.10 se ilustra, para cada sistema de ASV utilizado, dos gráficas. La gráfica superior es el histograma conjunto de puntuaciones de LO,LOS, segregado por sexo masculino. Se muestran dos distribuciones de densidad de frecuencia de puntuación. La distribución azul hace referencia al LO masculino y la distribución roja al LOS masculino. La gráfica inferior es la tasa de falsos positivos y falsos negativos en función del umbral de puntuación, bajo suplantación masculina. Ambas gráficas son funcionales para estudiar el rendimiento del sistema de ASV bajo suplantación de identidad masculina.

En la Figura 3.11 se ilustra, para cada sistema de ASV utilizado, dos gráficas. La gráfica superior es el histograma conjunto de puntuaciones de LO,LOS, segregado por sexo femenino. Se muestran dos distribuciones de densidad de frecuencia de puntuación. La distribución azul hace referencia al LO femenino y la distribución roja al LOS femenino. La gráfica inferior es la tasa de falsos positivos y falsos negativos en función del umbral de puntuación, bajo suplantación femenina. Ambas gráficas son funcionales para estudiar el rendimiento del sistema de ASV bajo suplantación de identidad femenina.

En la Figura 3.12 se ilustra, para cada sistema de ASV utilizado, dos gráficas. La gráfica superior es el histograma conjunto de puntuaciones de LO,LOS, agregado (locutores masculinos y femeninos). Se muestran dos distribuciones de densidad de frecuencia de puntuación. La distribución azul hace referencia al LO y la distribución roja al LOS. La gráfica inferior es la tasa de falsos positivos y falsos negativos en función del umbral de puntuación, bajo suplantación masculina y femenina. Ambas gráficas son funcionales para estudiar, de forma general, el rendimiento del sistema de ASV bajo suplantación independiente del sexo del locutor legítimo.

En la Figura 3.13 se representan las curvas DET asociadas a cada uno de los modelos ASV analizados, TDNN (x-vector), ResNet (r-vector) y ECAPA-TDNN. Se muestra el rendimiento del sistema de ASV cuando se realiza suplantación de identidad (independiente del sexo) mediante síntesis de voz, utilizando Tacotron 2 + HiFi-GAN. Adicionalmente se presenta la EER y AUC de cada sistema. El AUC DET, cuantifica la probabilidad de que el sistema cometa un error en su decisión. A mayor AUC, peor es el comportamiento del sistema ASV. La curva DET es muy representativa cuando se quieren comparar diferentes modelos. Se puede visualizar como el peor modelo es ECAPA, puesto que es el que mayor AUC y EER implica.

Se demuestra que el sistema de ASV basado en TDNN colleva el mismo rendimiento, independientemente del sexo del locutor. En cambio, en los modelos ResNet y ECAPA-TDNN si varía el rendimiento en función del sexo del locutor. En ResNet y ECAPA-TDNN el rendimiento es mejor cuando el locutor es femenino. La diferencia es de un 1% en ResNet, es decir, apenas varía el rendimiento, aunque es sustancialmente mejor cuando habla un locutor femenino. En contraposición, en ECAPA-TDNN, la diferencia del rendimiento según el sexo si es destacable. Cuando habla un locutor impostor masculino el sistema se ve doblemente perjudicado en robustez con respecto a cuando es de sexo femenino.

Analizando el comportamiento general (agregado), el modelo de ASV que mejor

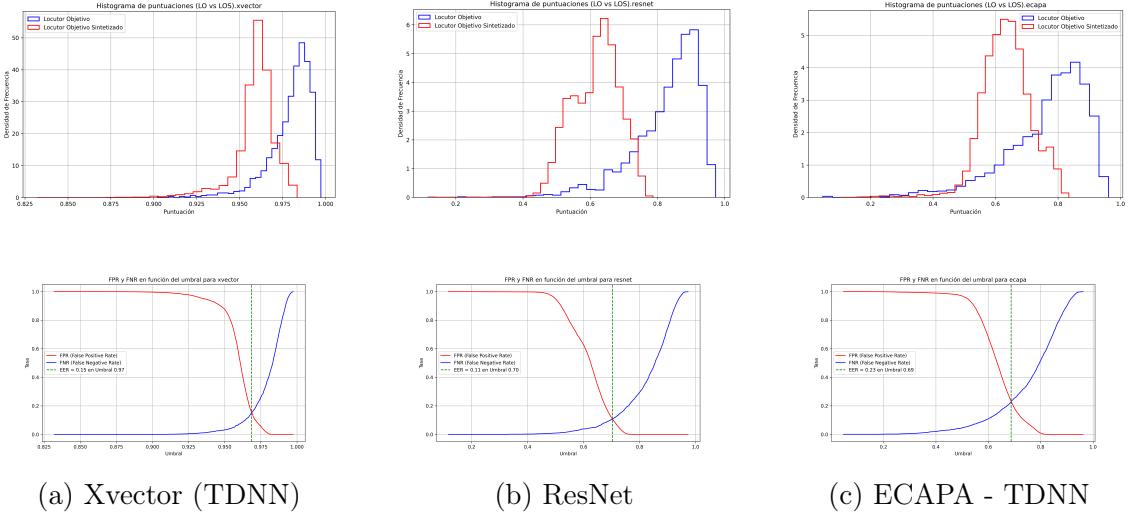


Figura 3.10: Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs umbral (Segregado por sexo masculino).

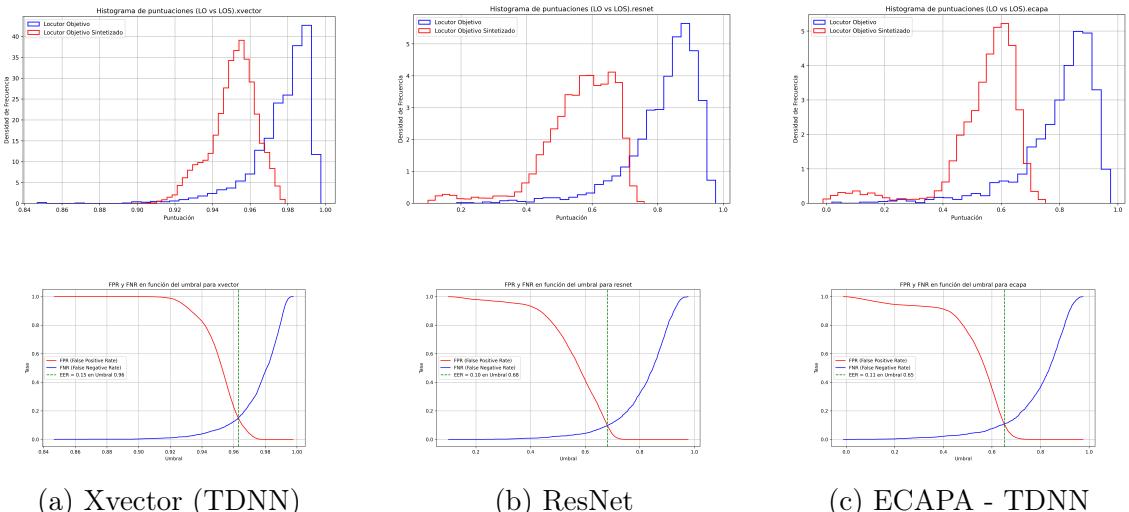


Figura 3.11: Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs umbral (Segregado por sexo masculino).

se comporta es ResNet, con una EER del 10,20 %. Le sigue x-vector con una EER del 15,47 % y finalmente, ECAPA-TDNN con una EER del 16,63 %. Esto indica que el sistema de ASV que mejor resiste los ataques de spoofing, específicamente en la síntesis de voz, es ResNet.

3.3.3. Resultado final

En la Figura 3.14 se ilustra una tabla donde aparecen las métricas de rendimiento asociadas a los diferentes experimentos anteriormente realizados. En este caso, se ilustra además de las métricas del rendimiento de referencia y el rendimiento en condiciones de suplantación, se representa la diferencia de ambos rendimientos, puesto que define la vulnerabilidad del sistema al ataque de suplantación en particular considerado.

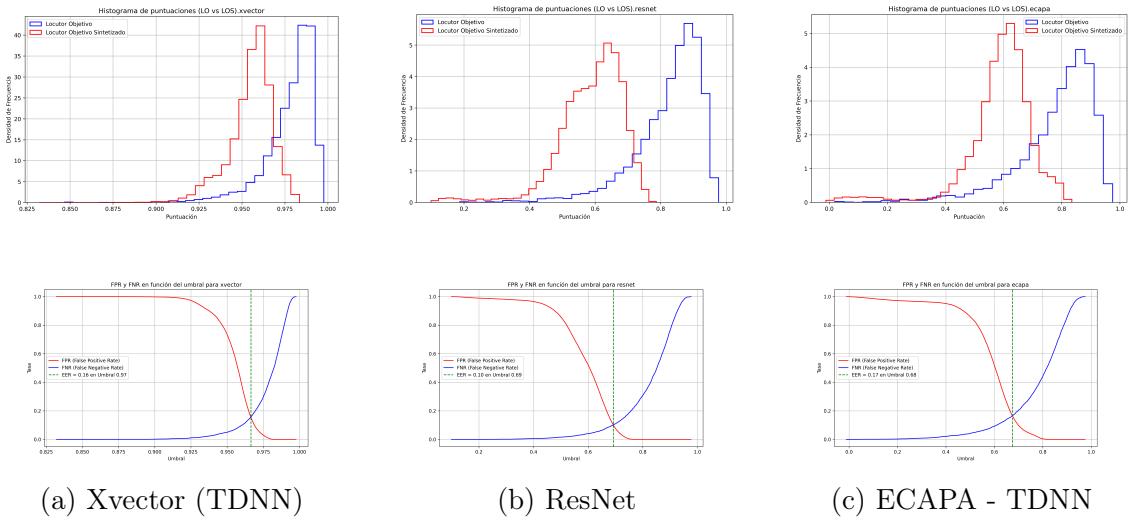


Figura 3.12: Histograma conjunto de puntuaciones (LO, LOS) y (FPR, FNR) vs umbral (Agregado).

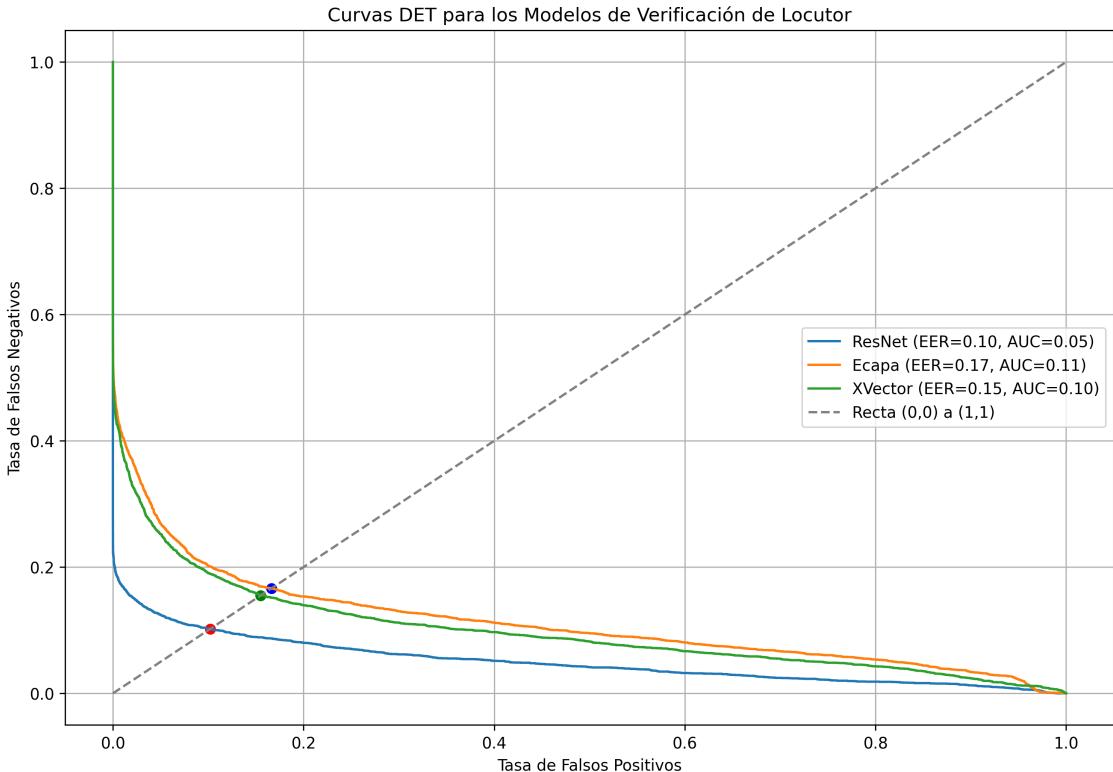


Figura 3.13: Curvas DET de los 3 sistemas de ASV analizados, cuando se realiza síntesis de voz utilizando Tacotron 2 + Diffwave.

Se deduce que el sistema más vulnerable al ataque de suplantación mediante síntesis de voz utilizando el sistema de TTS FastSpeech 2 + DiffWave es ECAPA-TDNN, puesto que conlleva la máxima EER relativa, con un valor del 12,13 %. El sistema más vulnerable al ataque de suplantación mediante síntesis de voz utilizando el sistema de TTS Tacotron 2 + HiFi-GAN es ECAPA-TDNN también, puesto que conlleva la máxima EER relativa, con un valor del 16,36 %.

Debido a que el 'enrollement' del locutor objetivo en el sistema de TTS Tacotron 2 + HiFi-GAN, denominado S_2 , se realizó con 100 sentencias diferentes, y el locutor objetivo del experimento mediante el sistema de TTS FastSpeech 2 + DiffWave, denominado S_1 , también se generó con la misma cantidad de sentencias, es posible comparar la calidad de la voz sintética proveniente de ambos sistemas. Sin embargo, para realizar una comparación fiel, sería necesario:

- Que las bases de datos utilizadas fueran similares en ambos sistemas, es decir, tanto la base de datos del locutor objetivo como la del locutor objetivo sintetizado. En el experimento con el sistema S_1 , el LO y LOS es único. En el experimento con el sistema S_2 , el LO y LOS está compuesto por 20 locutores. Esta diferencia en las bases de datos pueden generar métricas de rendimiento del sistema de ASV diferentes.
- Que se utilizara el mismo vocoder, puesto que los vocoders utilizados plantean arquitecturas diferentes. De esta manera se puede evaluar la calidad del sintetizador de voz, encargado de realizar la transformación del texto a una representación de características intermedias, típicamente el espectrograma Mel.

En la Tabla 3.8 se muestra una comparación de la EER de los sistemas de ASV: x-vector, ResNet y ECAPA-TDNN, para la suplantación de locutor mediante dos sistemas de TTS: FastSpeech 2 + DiffWave, denominado S_1 , utilizando 1 locutor objetivo sintetizado, y Tacotron 2 + HiFi-GAN, denominado S_2 , utilizando 20 locutores objetivo sintetizados. En el sistema de ASV x-vector, la EER es mayor en el sistema de TTS S_2 . En el sistema de ASV ResNet, la EER es mayor en el sistema de TTS S_1 , y en el sistema de ASV ECAPA-TDNN, la EER es mayor en el sistema de TTS S_1 . Debido a que una mayor EER se traduce en una menor precisión en la verificación del locutor por parte del sistema de ASV, se deduce que la mejor calidad de síntesis de voz se obtiene en el sistema de TTS FastSpeech 2 + DiffWave.

Esto posiblemente se deba a que el modelo FastSpeech 2 incorpora la arquitectura del Transformer, el cual es muy adecuado en tareas de conversión de texto [48]. Además, permite un control completo y preciso de todas las características prosódicas de la voz del locutor. Haciendo cambios incluso en la duración del carácter.

		S1 (FastSpeech 2 + DiffWave)				S2 (Tacotron 2 + HiFiGAN)	
	EER referencia [%]	EER suplantación [%] EP-1000	EER suplantación [%] EP-100	EER relativo [%] EP-1000	EER relativo [%] EP-100	EER suplantación [%]	EER relativo [%]
x-vector	2,43	3,90	3,90	1,47	1,47	15,47	13,04
ResNet	0,06	9,30	15,10	9,24	15,04	10,20	10,14
ECAPA-TDNN	0,27	12,40	18,30	12,13	18,03	16,63	16,36

Figura 3.14: Tabla asociada a las tasas de error equivalente (EERs) de los diferentes experimentos realizados.

Sistema ASV	S_1 (EER %) \times 1 LO	S_2 (EER %) \times 20 LO
x-vector	3,90	15,47
ResNet	15,10	10,20
ECPA-TDNN	18,30	16,63

Tabla 3.8: Comparación de la EER de los sistemas de ASV: x-vector, ResNet y ECAPA-TDNN, para la suplantación de locutor mediante dos sistemas de TTS: FastSpeech 2 + DiffWave, utilizando 20 locutores objetivos sintetizados, y Tacotron 2 + HiFi-GAN, utilizando 1 locutor objetivo sintetizado.

Capítulo 4

Conclusiones

Los sistemas de verificación automática de locutores (ASV) evaluados en este trabajo han demostrado ser vulnerables a los ataques de suplantación de identidad (spoofing) mediante síntesis de voz. La precisión de estos sistemas se redujo significativamente al emplear técnicas de spoofing, como se evidenció con los modelos de TTS utilizados basados en IA.

En el análisis del rendimiento de la precisión de referencia del sistema de ASV, los modelos ResNet y ECAPA-TDNN lograron una precisión en la verificación del locutor aproximadamente máxima, mientras que el sistema x-vector presentó una precisión del 97,57%, es decir, una EER del 2,43%.

Al generar voz sintética mediante el sistema de TTS FastSpeech 2 + DiffWave, la precisión del sistema de ASV se redujo con respecto a la precisión de referencia un 1,47%, 9,24% y un 12,13%, en los sistemas de ASV x-vector, ResNet y ECAPA-TDNN, respectivamente. Siendo el modelo ECAPA-TDNN el más afectado, con una precisión de verificación del locutor del 87,60%, es decir, una EER del 12,40%.

Al generar voz sintética mediante el sistema de TTS Tacotron 2 + HiFi-GAN, los sistemas de ASV x-vector, ResNet y ECAPA-TDNN experimentaron reducciones en la precisión, con respecto a la precisión de referencia, del 13,04%, 10,14% y 16,36%, respectivamente. El sistema ECAPA-TDNN fue nuevamente el más afectado, alcanzando una precisión del 83,37%, es decir, una EER del 16,63%. Estos resultados demuestran la efectividad de las tecnologías de TTS utilizadas para generar voces sintéticas convincentes que pueden comprometer la seguridad de los sistemas de verificación biométrica de voz.

El análisis indicó que la fase de inscripción en los sistemas ASV es crucial para proteger al locutor legítimo de ataques de suplantación de identidad. La complejidad de la inscripción afecta directamente al rendimiento del sistema. Los sistemas de ASV ResNet y ECAPA-TDNN mostraron una disminución en el rendimiento cuando la inscripción del locutor legítimo fue menos compleja, mientras que el sistema x-vector mantuvo una eficiencia constante independientemente de la complejidad de la inscripción, sugiriendo una mayor robustez en este aspecto.

El análisis dependiente del sexo del locutor reveló que el rendimiento de los sistemas ASV puede variar según el sexo del locutor en presencia de ataques de suplantación. Sin embargo, cuando no se realizaban ataques de suplantación, el rendimiento era similar, independientemente del sexo. Ante situaciones de suplantación, se concluyó que el sistema ResNet era apenas variable respecto al sexo del locutor, mostrando un rendimiento ligeramente mejor con voces femeninas (1% más de pre-

cisión). El sistema x-vector no mostró dependencia del sexo del locutor, mientras que el sistema ECAPA-TDNN tuvo una precisión notablemente mayor al verificar voces femeninas en comparación con voces masculinas. Esta disparidad resalta la necesidad de ajustar los modelos de ASV para minimizar los sesgos de género.

Se dedujo a partir de los resultados obtenidos en el rendimiento de los sistemas de ASV analizados, que el sistema de TTS FastSpeech 2 + DiffWave era mejor que el sistema de TTS Tacotron 2 + HiFi-GAN debido a ciertos factores, como que FastSpeech incorpora la arquitectura del Transformer, el cual es muy adecuado en tareas de conversión de texto y además permite un control completo y preciso de todas las características prosódicas de la voz del locutor. Sin embargo, para poder evaluar fielmente la comparación de la calidad de la síntesis mediante el sistema de ASV es necesario utilizar la misma cantidad de datos y preferiblemente los mismos locutores, siempre cumpliendo con el mismo criterio de inscripción para los diferentes locutores legítimos. En los experimentos, la comparación se realizó basándose en un proceso de inscripción del locutor legítimo equivalente en el sistema de ASV, sin embargo, utilizando bases de datos diferentes y con una cantidad de datos diferente.

Hubiera sido interesante comparar ambos sistemas de TTS utilizando una misma base de datos para extraer más información comparativa sobre la calidad de la síntesis de voz generada. Sin embargo, no existían modelos preentrenados para una base de datos común y, debido a la limitación de tiempo y a ciertos problemas técnicos, no se pudo realizar el entrenamiento de los modelos correspondientes.

La calidad de la síntesis se evaluó a partir del rendimiento de la precisión del sistema de ASV ante ataques de suplantación de identidad utilizando las síntesis correspondiente. Además, se describió cómo utilizando el mismo vocoder se podría analizar la calidad de los sintetizadores de voz, FastSpeech 2 y Tacotron 2 de forma independiente.

Además, para realizar un análisis exhaustivo de la precisión del sistema ASV, habría sido conveniente utilizar la misma cantidad de datos con las mismas sentencias tanto en la evaluación del rendimiento de referencia como en la evaluación del rendimiento en condición de suplantación. Es decir, que el locutor impostor y el locutor objetivo sintetizado reflejaran las mismas sentencias. También es importante que el locutor objetivo en ambas evaluaciones sea el mismo y que se utilice el mismo modelo de ASV.

Bibliografía

- [1] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370, 2012.
- [2] P. L. De Leon, M. Pucher, J. Yamagishi, I. Hernaez, and I. Saratxaga. Evaluation of speaker verification security and detection of hmm-based synthetic speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(8):2280–2290, 2012.
- [3] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4685–4694, 2019.
- [4] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification. In *Proc. Interspeech 2020*, pages 3830–3834, 2020.
- [5] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. Ecpa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. In *Interspeech 2020, interspeech_2020.ISCA, October 2020*.
- [6] V. Dumoulin and F. Visin. Una guía de aritmética de convolución para el aprendizaje profundo. <https://arxiv.org/pdf/1603.07285.pdf>, 2016. Preimpresión de arXiv:1603.07285.
- [7] M. Faundez-Zanuy and E. Monte-Moreno. State-of-the-art in speaker recognition. *IEEE Aerospace and Electronic Systems Magazine*, 20(5):7–12, May 2005.
- [8] Zhifu Gao, Yan Song, Ian McLoughlin, Pengcheng Li, Yiheng Jiang, and Li-Rong Dai. Improving Aggregation and Loss Function for Better Embedding Learning in End-to-End Speaker Verification System. In *Proc. Interspeech 2019*, pages 361–365, 2019.
- [9] L. Paola Garcia-Perera, Juan A. Nolazco-Flores, Bhiksha Raj, and Richard Stern. Optimization of the det curve in speaker verification. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 318–323, 2012.
- [10] Hugo Garzón, Jimy Cortes, and José Chaves Osorio. Del análisis de fourier a las wavelets - transformada continua wavelet (cwt). *Scientia Et Technica*, 01 2007.
- [11] A. Gibiansky, S. O. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou. Deep voice 2: Multi-speaker neural text-to-speech. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2017.

- [12] Y. González, H. Juárez, O. Rocha, R. Hernández, and A. Bermúdez. Evaluación comparativa de sistemas de reconocimiento de locutor basados en los algoritmos lpc, cc y mfcc. *Memoria Investig. Ing. (Facultad Ing., Univ. Montev.)*, (17):121–136, November 2019.
- [13] D. W. Griffin and J. S. Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 236–243, 1984.
- [14] Richard Hahnloser, Rahul Sarpeshkar, Misha Mahowald, Rodney Douglas, and H. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–51, 07 2000.
- [15] R. G. Hautamäki, T. Kinnunen, V. Hautamäki, T. Leino, and A.-M. Laukkanen. I-vectors meet imitators: On vulnerability of speaker verification systems against voice mimicry. In *Interspeech*, pages 930–934. Citeseer, 2013.
- [16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv*, 2015.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [18] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.
- [19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, pages 448–456, 2015.
- [20] A. Kanagasundaram, D. Dean, R. Vogt, M. McLaren, S. Sridharan, and M. Mason. Weighted lda techniques for i-vector based speaker verification. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4781–4784, Kyoto, Japan, 2012.
- [21] A. Karpathy. Cs231n: Redes neuronales convolucionales para el reconocimiento de la visión. <http://cs231n.github.io/convolutional-networks/>, 2017.
- [22] P. Kenny, G. Boulian, P. Ouellet, and P. Dumouchel. Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15:1435–1447, 2007.
- [23] T. Kinnunen, Z.-Z. Wu, K. A. Lee, F. Sedlak, E. S. Chng, and H. Li. Vulnerability of speaker verification systems against voice conversion spoofing attacks: The case of telephone speech. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4401–4404. IEEE, 2012.
- [24] J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *arXiv preprint arXiv:2010.05646*, 2020.
- [25] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis, 2021.
- [26] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C. Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. In

Advances in Neural Information Processing Systems, volume 32, pages 14910–14921, 2019.

- [27] Y. Kumar, A. Koul, and C. Singh. A deep learning approaches in text-to-speech system: A systematic review and recent research perspective. *Multimedia Tools and Applications*, 82(10):15171–15197, 2023.
- [28] K. Lee and R. V. Cox. A very low bit rate speech coder based on a recognition/synthesis paradigm. *IEEE Transactions on Speech and Audio Processing*, 9:482–491, 2001.
- [29] K. A. Lee, O. Sadjadi, H. Li, and D. Reynolds. Two decades into speaker recognition evaluation - are we there yet? *Computer Speech Language*, 61:101058, 2020.
- [30] Kong Aik Lee, Ville Vestman, and Tomi Kinnunen. Asvtorch toolkit: Speaker verification with deep neural networks. *SoftwareX*, 14:100697, 2021.
- [31] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding, 2017.
- [32] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, pages 498–502, 2017.
- [33] A. McCree. Multiclass discriminative training of i-vector language recognition. In *Proc. Odyssey*, pages 166–172, 2014.
- [34] L. C. Oliveira, M. C. Viana, and I. M. Trancoso. A rule-based text-to-speech system for portuguese. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 73–76. IEEE Computer Society, March 1992.
- [35] A. V. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, and A. Graves. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [37] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Proc. Interspeech 2018*, pages 3743–3747, 2018.
- [38] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, K. Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý. The kaldi speech recognition toolkit. In *Proceedings of ASRU 2011*, pages 1–4. IEEE Signal Processing Society, 2011.
- [39] PyWorld Vocode Project. Página web oficial de descarga de pyworldvocoder. <https://pypi.org/project/pyworld/0.3.1/>, 2023.
- [40] Lawrence R. Rabiner and Ronald W. Schafer. Introduction to digital speech processing. *Foundations and Trends® in Signal Processing*, 1(1–2):1–194, 2007.

- [41] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [42] D. Raj, D. Snyder, D. Povey, and S. Khudanpur. Probing the information encoded in x-vectors. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 726–733. IEEE, December 2019.
- [43] Mirco Ravanelli, Titouan Parcollet, Adel Moumen, Sylvain de Langen, Cem Subakan, Peter Plantinga, Yingzhi Wang, Pooneh Mousavi, Luca Della Libera, Artem Ploujnikov, Francesco Paissan, Davide Borra, Salah Zaiem, Zeyu Zhao, Shucong Zhang, Georgios Karakasidis, Sung-Lin Yeh, Pierre Champion, Aku Rouhe, Rudolf Braun, Florian Mai, Juan Zuluaga-Gomez, Seyed Mahed Mousavi, Andreas Nautsch, Xuechen Liu, Sangeet Sagar, Jarod Duret, Salima Mdhaffar, Gaelle Lapierre, Mickael Rouvier, Renato De Mori, and Yannick Esteve. Open-source conversational ai with speechbrain 1.0, 2024.
- [44] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, Elena Rastorgueva, François Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, and Yoshua Bengio. SpeechBrain: A general-purpose speech toolkit, 2021. arXiv:2106.04624.
- [45] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech, 2022.
- [46] D. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1):19–41, 2000.
- [47] D. A. Reynolds and W. M. Campbell. Text-independent speaker recognition. In J. Benesty, M. M. Sondhi, and Y. A. Huang, editors, *Springer Handbook of Speech Processing*, Springer Handbooks, pages 763–780. Springer, Berlin, Heidelberg, 2008.
- [48] Markéta Řezáčková, Jan Švec, and Daniel Tihelka. T5g2p: using text-to-text transfer transformer for grapheme-to-phoneme conversion. 2021.
- [49] E. Rosello, A. Gomez-Alanis, A. M. Gomez, and A. Peinado. A conformer-based classifier for variable-length utterance processing in anti-spoofing. In *Proc. Interspeech 2023*, pages 5281–5285, 2023.
- [50] M. Senoussaoui, P. Kenny, N. Dehak, and P. Dumouchel. An i-vector extractor suitable for speaker recognition with both microphone and telephone speech. In *Odyssey*, page 6, June 2010.
- [51] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, 2018.
- [52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
- [53] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur. Deep neural network

- embeddings for text-independent speaker verification. In *Proceedings of Interspeech*, pages 999–1003, 2017.
- [54] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur. Speaker recognition for multi-speaker conversations using x-vectors. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5796–5800. IEEE, May 2019.
 - [55] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333. IEEE, April 2018.
 - [56] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, 2018.
 - [57] David Snyder, Pegah Ghahremani, Daniel Povey, Daniel Garcia-Romero, Yishay Carmiel, and Sanjeev Khudanpur. Deep neural network-based speaker embeddings for end-to-end speaker verification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 165–170, 2016.
 - [58] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Proc. NIPS*, pages 3104–3112, 2014.
 - [59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.
 - [60] Pin Shen Teh, Ning Zhang, Andrew Beng Jin Teoh, and Ke Chen. A survey on touch dynamics authentication in mobile devices. *Computers Security*, 59:210–235, 2016.
 - [61] Sreenivas Sremath Tirumala, Seyed Reza Shahamiri, Abhimanyu Singh Garhwal, and Ruili Wang. Speaker identification features extraction methods: A systematic review. *Expert Systems with Applications*, 90:250–271, 2017.
 - [62] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.
 - [63] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4052–4056, 2014.
 - [64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
 - [65] J. Villalba and E. Lleida. Detecting replay attacks from far-field recordings on speaker verification systems. In *Biometrics and ID Management: COST 2101 Eu-*

ropean Workshop, BioID 2011, Brandenburg (Havel), Germany, March 8–10, 2011. Proceedings 3, pages 274–285. Springer, 2011.

- [66] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Leibny Paola García-Perera, Fred Richardson, Réda Dehak, Pedro A. Torres-Carrasquillo, and Najim Dehak. State-of-the-art speaker recognition with neural network embeddings in nist sre18 and speakers in the wild evaluations. *Computer Speech Language*, 60:101026, 2020.
- [67] A. Waibel. Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1(1):39–46, 1989.
- [68] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang. Community preserving network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
- [69] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, et al. Tacotron: Towards end-to-end speech synthesis, 2017. arXiv preprint arXiv:1703.10135.
- [70] C. W. Wightman and D. T. Talkin. The aligner: Text-to-speech alignment using markov models. In *Progress in Speech Synthesis*, pages 313–323. Springer New York, 1997.
- [71] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li. Spoofing and countermeasures for speaker verification: A survey. *Speech Communication*, 66:130–153, 2015.
- [72] Z. Wu, A. Khodabakhsh, C. Demiroglu, J. Yamagishi, D. Saito, T. Toda, and S. King. Sas: A speaker verification spoofing database containing diverse attacks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4440–4444. IEEE, 2015.
- [73] Zhizheng Wu, Nicholas Evans, Tomi Kinnunen, Junichi Yamagishi, Federico Alegre, and Haizhou Li. Spoofing and countermeasures for speaker verification: A survey. *Speech Communication*, 66:130–153, 2015.
- [74] R. Yamamoto, E. Song, and J.-M. Kim. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203, 2020.
- [75] H. Zen and H. Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4470–4474, South Brisbane, QLD, Australia, 2015. IEEE.