ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Grado en Ingeniería en Tecnologías de Telecomunicación

Evaluación de clasificadores de voz para Smart Personal Assistants

Autor

Pablo Ríos Goytre

Dirigido por

Gregorio López López

Roberto Gesteira Miñarro

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Evaluación de clasificadores de voz para Smart Personal Assistants

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico **2022/23** es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Pablo Ríos Goytre

Fdo.: ..... Fecha: ...../ ...../ .....

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Gregorio López López

Fdo.: ..... Fecha: ...../ ...../ .....

Roberto Gesteira Miñarro

Roberto Gesteira Miñarro

GESTEIRA MIÑARRO ROBERTO - 48228840W

Firmado digitalmente por GESTEIRA MIÑARRO ROBERTO - 48228840W
Fecha: 2023.07.04 12:35:59 +02'00'

Fdo.: ..... Fecha: ...../ ...../ .....

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Grado en Ingeniería en Tecnologías de Telecomunicación

Evaluación de clasificadores de voz para Smart Personal Assistants

Autor

Pablo Ríos Goytre

Dirigido por

Gregorio López López

Roberto Gesteira Miñarro

Resumen

Palabras clave

Asistentes personales, Seguridad, Privacidad, Inteligencia Artificial, Redes Neuronales

Introducción

Los asistentes personales o Smart Personal Assistants (SPA) cuentan con una presencia cada vez más grande en los hogares y a medida que la tecnología avanza, más funcionalidades adquiere y más se extienden sus capacidades. Esto también hace a estos sistemas más susceptibles de recibir ataques hacia su seguridad, aprovechando posibles brechas que aparezcan y por lo tanto, hacer que tanto datos como aplicaciones de los asistentes puedan estar comprometidos.

Recientemente han aparecido nuevas amenazas como las herramientas de clonación de voz, en las cuales, a partir de una serie de grabaciones de un usuario y el uso de redes de Inteligencia Artificial, se puede crear una réplica de la voz original con una gran similitud. Esto, evidentemente supone un peligro real para los sistemas de autenticación de los asistentes personales, ya que en caso de que dichas réplicas puedan traspasar la seguridad, los datos se verán completamente expuestos mediante ataques que puede realizar cualquier persona ya que este tipo de plataformas de replicación de voz suelen ser de carácter público en su mayoría.

Debido a lo comentado anteriormente, los sistemas de seguridad de los SPA necesitan ser lo más fiables posibles y los sistemas de verificación de voces han de tener una gran precisión ya que ataques como la suplantación de voz mediante una réplica y otros tipos de métodos, pueden suponer una amenaza seria para estos sistemas. Este tipo de asistentes no suele contar precisamente con sistemas de seguridad férreos debido a que priorizan la velocidad de respuesta por delante de la protección de datos, por lo que es posible que se requiera de una revisión de los procesos de seguridad presentes actualmente.

Definición del proyecto

Este trabajo de fin de grado tiene como principal misión realizar un estudio de la seguridad de los SPA y valorar su protección frente a ataques de suplantación por voz. Además se valorará cómo de comprometidos están los datos en los asistentes más utilizados. Por otra parte se desarrollará un modelo de redes neuronales para tratar de clasificar voces reales y voces sintéticas, comprobando si estas redes son una herramienta recomendable para reforzar los sistemas de seguridad de los SPA.

En primer lugar, se realizarán una serie de pruebas para evaluar la protección de los asistentes personales ante distintos ataques de suplantación de identidad. Además, se valorará la dificultad de acceder a datos comprometidos como son aplicaciones financieras o sistemas de mensajería.

Además, se desarrollarán distintos modelos empleando redes neuronales convolucionales con el objetivo de clasificar voces reales y voces generadas por ordenador. Se evaluará el desempeño de los modelos y se demostrará si son indicadas para implementarse en los sistemas de seguridad de los asistentes personales.

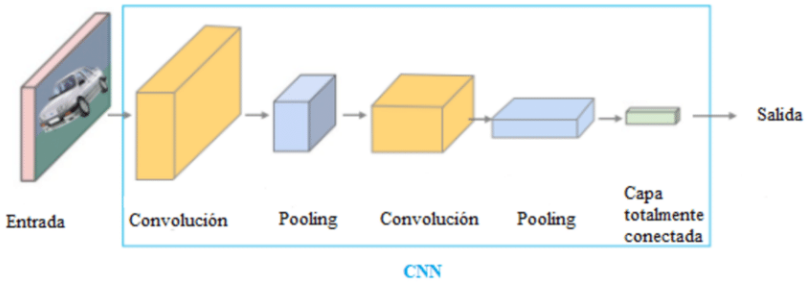

Ilustración 1. Esquema de una red neuronal convolucional

Resultados y conclusiones

Las pruebas de seguridad realizadas a los SPA mostraron que los sistemas de autenticación con los que cuentan son frágiles y fácilmente esquivables. Por una parte, se demostró que todos los asistentes personales testeados interpretan las réplicas de voces generadas por Inteligencia Artificial como la propia voz a la que suplanta, lo que indica que se requiere de un avance urgente en sistemas de diferenciación de este tipo de voces. A la hora de valorar la complejidad del acceso a posibles datos comprometidos, todos los asistentes empleados para estas pruebas han mostrado la existencia de una o varias brechas de seguridad en este aspecto, otorgando acceso a aplicaciones con posible información privada sin requerir de ninguna capa adicional de seguridad aparte de la identificación inicial.

Los resultados de los modelos de redes neuronales desarrollados para clasificar voces reales y voces sintéticas ofrecieron peores valores de lo esperado. La precisión del mejor modelo no superó el 70% en los datos de testing, aunque a pesar de ello, los resultados sí que eran realmente buenos en el proceso de entrenamiento.

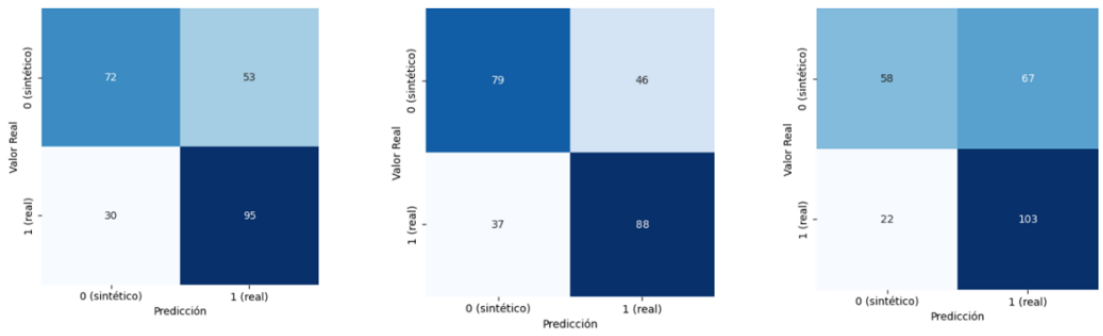

Ilustración 2. Matrices de confusión de los modelos desarrollados más precisos

A pesar de que los resultados de precisión no son perfectos, se ha demostrado que las redes neuronales convolucionales son una buena aproximación a este tipo de problemas y se ha de investigar en un futuro posibles mejoras a implementar para obtener cada vez mejores herramientas de clasificación de voces reales y sintéticas.

Abstract

Keywords

Smart Personal Assistants, Security, Privacy, Artificial Intelligence, Neural Networks

Introduction

Personal assistants or Smart Personal Assistants (SPAs) have an ever-growing presence in homes and as technology advances, the more functionalities it acquires and the more its capabilities extend. This also makes these systems more susceptible to attacks on their security, taking advantage of possible breaches that appear and therefore, making both data and applications of the assistants can be compromised.

Recently, new threats have appeared, such as voice cloning tools, in which, from a series of recordings of a user and the use of Artificial Intelligence networks, a replica of the original voice can be created with a great similarity. This obviously poses a real danger to the authentication systems of personal assistants, since in the event that these replicas can breach security, the data will be completely exposed through attacks that can be carried out by anyone, since this type of voice replication platforms are usually public in most cases.

Due to the above, SPA security systems need to be as reliable as possible and voice verification systems must be highly accurate, since attacks such as voice impersonation through replication and other types of methods can pose a serious threat to these systems. These types of assistants do not usually have strong security systems because they prioritize speed of response over data protection, so a review of the security processes currently in place may be required.

Project Definition

The main purpose of this final degree project is to carry out a study of the security of SPAs and to evaluate their protection against voice impersonation attacks. In addition, it will be evaluated how compromised the data in the most used assistants are. On the other hand, a neural network model will be developed to try to classify real voices and synthetic voices, checking if these networks are a recommendable tool to reinforce SPA security systems.

First, a series of tests will be carried out to evaluate the protection of personal assistants against different impersonation attacks. In addition, the difficulty of accessing compromised data such as financial applications or messaging systems will be assessed.

In addition, different models will be developed using convolutional neural networks with the aim of classifying real voices and computer-generated voices. The performance of the models will be evaluated, and it will be demonstrated whether they are suitable for implementation in personal assistant security systems.

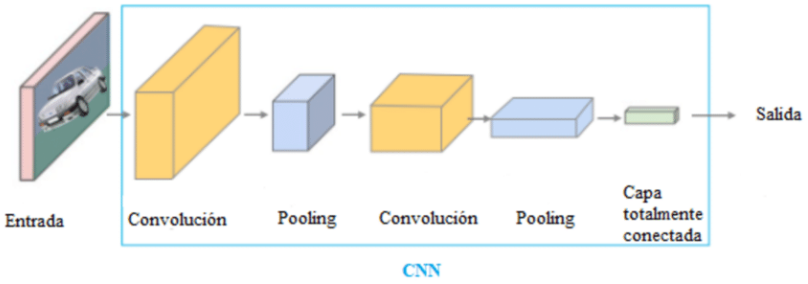

Illustration 3. Schematic diagram of a convolutional neural network

Results and conclusions

The security tests performed on the SPAs showed that their authentication systems are fragile and easily circumvented. On the one hand, it was shown that all the personal assistants tested interpret the voice replicas generated by Artificial Intelligence as the voice itself, which indicates that urgent progress is required in systems for differentiating this type of voice. When assessing the complexity of access to possible compromised data, all the assistants used for these tests have shown the existence of one or more security gaps in this aspect, granting access to applications with possible private information without requiring any additional layer of security apart from the initial identification.

The results of the neural network models developed to classify real voices and synthetic voices offered worse values than expected. The accuracy of the best model did not exceed 70% on the test data, although the results were good in the training process.

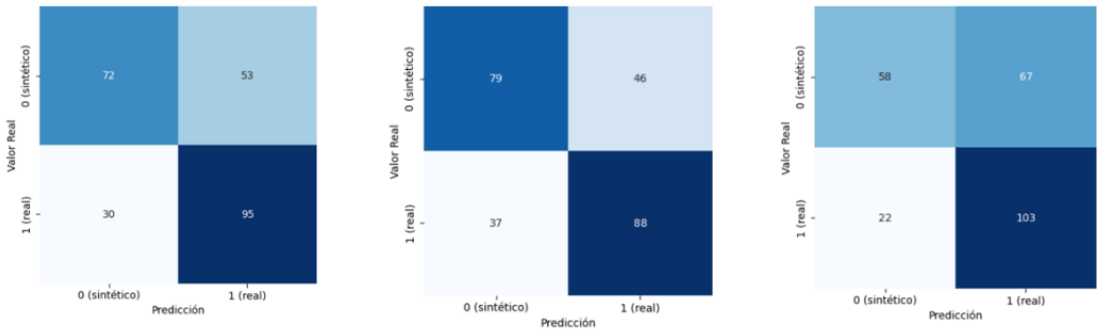

Illustration 4. Confusion matrices of the most accurate models developed.

Although the accuracy results are not perfect, it has been demonstrated that convolutional neural networks are a good approximation to this type of problems and possible improvements to be implemented in the future should be investigated in order to obtain better and better tools for the classification of real and synthetic voices.

Índice general

Índice de figuras

Índice de tablas

Índice de fragmentos de código

Capítulo 1. Introducción

Este trabajo de fin de grado tendrá como principal misión el análisis de los sistemas de seguridad de los asistentes personales inteligentes o Smart Personal Assistants (SPA) ante ataques de suplantación de voz y la propuesta de clasificadores para distinguir entre voces reales y voces sintéticas para poder reforzar la protección de los asistentes ante este tipo de amenazas. Hoy en día estos dispositivos cada vez están más presentes en los hogares y a medida que se desarrolla la tecnología, más funcionalidades adquiere y se extienden sus posibilidades. Esto también implica un aumento del número de posibles brechas de seguridad y por lo tanto, ahora más que nunca es indispensable reforzar los sistemas de seguridad y privacidad para no sufrir ataques y poder utilizar la tecnología sin peligro [1].

En un primer lugar se realizarán una serie de pruebas para comprobar cómo de protegidos están los asistentes más vendidos frente a ataques de suplantaciones de identidad por medio de diversos métodos. Además, se comprobará la complejidad a la hora de acceder a aplicaciones y datos con contenido comprometido como pueden ser aplicaciones de bancas y finanzas o sistemas de mensajería.

En este mismo contexto, se elaborarán una serie de modelos de redes neuronales con el objetivo de poder diferenciar voces reales y voces sintéticas. Se analizará el desempeño de estos modelos y se valorará si pueden ser sistemas aplicables a los SPA para reforzar su seguridad respecto a ataques mediante sistemas de clonación de voces, los cuales han incrementado su popularidad recientemente.

1.1. Motivación

A pesar de los avances en los sistemas de seguridad en los dispositivos actuales, estos no atacan el problema cada vez más común de suplantación de voces, por lo que es indispensable añadir una nueva capa de seguridad. Si bien los sistemas de verificación y autenticación actuales pueden ser útiles y refuerzan sin lugar a dudas la privacidad y seguridad de los asistentes personales, en este trabajo se buscarán alternativas para atacar directamente estos nuevos problemas.

Los sistemas utilizados para la clasificación de voces de los asistentes han de ser renovados continuamente ya que cada vez hay más recursos para explorar. Un sistema con tanto potencial como los clasificadores de voces debe ser una tecnología que mejore exponencialmente, ya no solo por su aplicación en los SPA, sino por su extensa presencia en todo tipo de dispositivos en la actualidad, así como su más que probable expansión hacia otros en los que no se encuentre actualmente.

Como veremos adelante, uno de los principales alicientes para perfeccionar estos programas son los algoritmos de Deep Learning que van apareciendo y mejorando cada vez más, así como una cantidad de datos prácticamente infinitos conteniendo voces y recursos para poder entrenar los modelos que se hagan. Estos recursos serán de gran ayuda para atacar los problemas mencionados, y será de interés ver como responden los sistemas ya establecidos a nuevos y distintos inputs.

1.2. Objetivos del proyecto

Los objetivos del proyecto son:

- Estudio de los sistemas de biometría y autenticación utilizados por los asistentes personales.
- Análisis de seguridad de asistentes personales ante ataques de suplantación de identidad.
- Análisis de seguridad y privacidad en el acceso a datos comprometidos mediante asistentes personales.
- Desarrollo de modelos basados en redes neuronales para clasificar voces reales y sintéticas
- Estudio de herramientas para mejorar los modelos desarrollados.

Capítulo 2. Estado del Arte

2.1. Seguridad en los Smart Personal Assistants

Los asistentes personales además de ofrecer numerosos beneficios en la vida cotidiana también plantean cierta preocupación acerca de su seguridad y privacidad. Debido a fallos de diseño presentes en este tipo de asistentes, estos son más propensos a tener problemas de seguridad y compartir información privada. Los problemas a considerar son los siguientes:

- Recopilación de datos: Los SPAs almacenan una gran cantidad de información cada vez que son utilizados. Véase consultas, ubicaciones, mensajes, registros en los calendarios, todos estos elementos son guardados por el sistema para su posterior uso. Además, para reconocer las palabras de activación que les permite empezar a procesar consultas del cliente, estos dispositivos están en un modo de permanente escucha, y a pesar de que las empresas alegan que no se almacenan grabaciones que no sean estrictamente peticiones, han aparecido casos que demuestran lo contrario [2].

- Escucha accidental: En la misma línea del punto anterior, cabe mencionar la posibilidad de que el asistente interprete erróneamente una palabra o frase como el comando de activación y por lo tanto empiece a grabar audio sin que el usuario lo sepa.

- Seguridad en dispositivos conectados: Al contar con la posibilidad de controlar otros sistemas en el hogar, como cerraduras o luces, estos también se pueden ver amenazados en caso de posibles ciberataques.

Como se puede observar, la naturaleza abierta de los SPAs para facilitar la comunicación entre el mismo y el usuario puede abrir la puerta a posibles problemas de seguridad que pueden afectar a datos privados y dispositivos conectados del hogar. En el caso de que un actor malicioso decida intentar atacar a estos dispositivos, es probable que finalmente pueda acceder a los datos de este y poner en compromiso la privacidad de los usuarios.

2.2. Biometría de voz y posibles ataques

Para identificar la voz de cada individuo los asistentes personales hacen uso de la tecnología denominada como biometría de voz. Esta trata la voz de la persona como una característica única a la hora de autenticarla y permite un acceso seguro y rápido a los sistemas correspondientes. Números avances en las redes neuronales en los últimos años ha permitidos desarrollar esta tecnología para hacerla cada vez más precisa y eficaz requiriendo cada vez menos longitud de muestra de voz del individuo a autenticar.

En torno a 70 partes del cuerpo están involucradas en la voz de una persona. La biometría de voz hace uso de estas particularidades de cada persona y confía en una fuerte correlación entre ellas y cómo una persona habla. Mediante la extracción de estas características se extrae una "huella dactilar" en forma de voz, a partir de un proceso en el cual, a partir de una primera muestra inicial, se piden más muestras para mezclarlas y ajustarlas y finalmente crear un modelo de voz que identificará a la persona con precisión [3].

Una vez se ha creado este modelo a partir sonidos, palabras y frases habladas por la persona y transformados estos en señales eléctricas y a su vez codificándose se implementará en el sistema a utilizar. El modelo de voz en cuestión habrá identificado características como el ritmo y las pausas a la hora de hablar, el tono y distintos patrones de la voz que permitirá identificarla individualmente y asociarla a una persona [4].

Para verificar una nueva muestra de voz y averiguar de quién se trata se tomará una nueva muestra de voz, se mandará al sistema de biometría de voz, creará un nuevo modelo de voz a partir de la muestra recibida y verificará si esta voz coincide en características a alguna de las ya guardadas. El uso de estos sistemas de biometría de voz es cada vez más extenso debido a las mejoras en precisión, principalmente causado por los avances en el campo de inteligencia artificial que han permitido hacer de este sistema uno de los más comunes en cuanto a seguridad de protección de datos, siendo puestos cada vez más en duda herramientas tradicionales como el uso de contraseñas o patrones.

A raíz del uso de este tipo de sistemas de reconocimiento de voz, se han visto en aumento a su vez los posibles ataques que pueden sufrir los dispositivos que cuenten con este tipo de seguridad. Algunos de los más comunes son:

- Voice Squatting: consiste en un ataque en el cual el autor aprovecha la forma en la que una acción es llamada. En estos casos el atacante puede implementar una aplicación maliciosa que se invoque a raíz de una serie de palabras que puedan sonar muy parecidas a un comando normal del dispositivo. Una vez se pronuncien estos comandos, la aplicación maliciosa puede ser activada sin que el usuario se haya podido dar cuenta. A partir de esto, según cómo se haya desarrollado el programa para el ataque puede, o bien hacerse pasar por otras aplicaciones o decir comandos falsos de despedida ("Adiós", "Hasta luego") y permanecer en escucha para recopilar información durante un tiempo determinado [5].

- Ataques de suplantación de voz: Como se ha comentado, los sistemas de reconocimiento de voz, aunque ofrecen por lo general un buen nivel de precisión a la hora de reconocer voces, pueden sufrir ataques por parte de terceros que quieran hacerse pasar por el usuario original. Por una parte, se pueden producir ataques por medio de grabaciones de voz del usuario, en cuyo caso simplemente sería necesario contar con las palabras de activación correspondientes a cada dispositivo, ya que en su mayoría una vez se reconocen estas palabras iniciales, ya se tiene acceso a todos los datos del sistema [6]. Por otra parte, estos intentos de suplantación pueden producirse por medio de voces recreadas por inteligencia artificial que traten de recrear la del usuario original y permita reproducir las palabras que se quiera con esa voz y por lo tanto, poder pronunciar los comandos de activación mencionados.

2.3. Ataques con clonación de voces por IA

La clonación de voces mediante el uso de inteligencia artificial ha generado un gran interés en los últimos años debido a sus posibles usos e implicaciones en el mundo de la ciberseguridad. Consiste en la generación de una voz artificial que trata de parecerse y actúa de manera similar a una real y la cual una vez creada, puede decir lo que se quiera e incluso transmitir distintas emociones según se vea. Con unos segundos o minutos de una voz grabada, estos programas

pueden crear una muestra de entrenamiento para preparar un modelo que replique la voz y pueda leer cualquier texto.

Gracias al uso de redes neuronales basadas en Text-To-Speech, estas reconocen patrones y características de los datos para poder clasificar las voces. La velocidad y entonación de la voz son algunas de estas características que el modelo emplea para que el resultado sea una voz lo más humana posible [7]. La calidad de estas réplicas de voces cada vez está mejorando más y la longitud del input de voz original cada vez es menor como se puede ver en la herramienta Vall-E desarrollada por un equipo de Microsoft, la cual utiliza simplemente 3 segundos de audio para sintetizar y crear un nuevo modelo de voz, además de permitir recrear sentimientos y sonoridades [8].

Estos nuevos sistemas se están empezando a utilizar por parte de una nueva ola de cibercriminales, los cuales han visto en estos programas, una nueva herramienta para estafas y suplantación de voces. Estudios de McAfee aseguran que un cuarto de sus encuestados han sufrido ataques de clonación de voz o conocen a alguien que lo ha sufrido [9]. Los atacantes crean mensajes suplantando voces y pueden suponer un peligro grave en diversos campos. Por una parte, a mayor escala, estas suplantaciones pueden dar lugar a noticias falsas o audios manipulados que pueden afectar a la opinión pública sobre un tema, promover campañas de difamación o manipular a altos ejecutivos y dirigentes. Por otro lado, también supone un riesgo real a la hora de usarse como pruebas falsas para casos criminales. La diferenciación entre este tipo de voces y las reales es indispensable para no perjudicar equivocadamente una sentencia y hacer creer falsos testimonios. Posibles chantajes y ataques también pueden ser provocados debido al mal uso de estas aplicaciones. También pueden realizarse ataques de phishing que hagan creer a la víctima que se está comunicando con alguien con el que confía. En lo respectivo a ataques a asistentes personales, como ya se ha comentado, pueden verse replicadas biometrías de voz que permitan acceder a estos dispositivos suplantando al usuario original.

2.4. Clasificación de voces reales y voces con IA

Como se ha podido ver, las posibilidades de ataques mediante aplicaciones de clonación de voz cada vez van en aumento y es necesario para el futro de la ciberseguridad elaborar herramientas

al respecto. Para ello, las redes neuronales y el Deep Learning están trazando una nueva ruta a seguir a la hora de atacar este problema, desarrollando algoritmos y modelos que permiten clasificar este tipo de elementos y poder diferenciar aquellas voces pertenecientes a una persona real de una generada por un ordenador.

Con los ataques de voces generados por IA en aumento, y una constante evolución de estas tecnologías se hace indispensable desarrollar sistemas que permitan diferenciar voces reales de falsas lo antes posible [10]. Cuanto antes se desarrollen estos sistemas de seguridad, más probable es que los ataques se reduzcan o al menos sus efectos, y por ende, se puede evitar que muchas personas se vean perjudicadas y estafadas y puedan verse amenazados sus datos personales o puedan ser víctimas de robos monetarios o de bienes.

Para lograrlo, se exploran diferentes técnicas y enfoques utilizados en el procesamiento de señales de audio y en el análisis de características acústicas relevantes. Mediante la extracción de características discriminativas y diseños de aprendizaje automático en base a ellas, se construyen redes que permiten abordar este tipo de problemas de clasificación que permitan a un sistema identificar correctamente al usuario real y no dar lugar a posibles ataques de suplantación.

2.4.1. Extracción de características

Para realizar la clasificación de voces reales y voces generadas por ordenador se determinarán una serie de elementos a partir de las voces que permiten observar diferencias entre las dos clases y poder generar un modelo en torno a estas. Se extraerán imágenes a raíz de audios con las que se entrenará una red neuronal. Los principales componentes a extraer en este tipo de problemas suelen ser los siguientes:

- Espectrogramas: Muestran una representación visual del sonido mediante una serie de ondas. Representan cuál es el nivel de frecuencias del audio a lo largo de la duración del mismo. El comportamiento de las ondas es diferente entre voces reales y voces artificiales, así como las pausas son menos naturales y menos abruptas en el caso de estos últimos [11] [Figura 1]..

- Mel- Frequency Cepstral Coefficients (MFCC) son coeficientes que representan el habla humana basado en la percepción audible de la persona. Son habitualmente utilizados para el reconocimiento de voz [12].

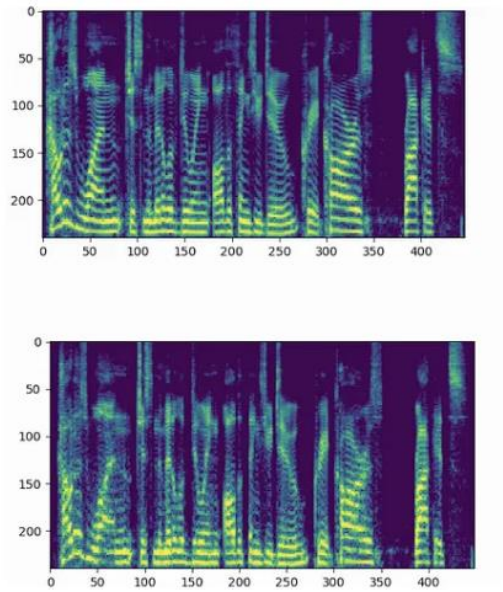

Figura 1. Comparativa de espectrograma de voz real (arriba) y de voz artificial (abajo) [9]

- Centroide espectral: Se trata de una medida empleada para definir un espectro. Localiza e indica a qué frecuencia se centra la energía de un espectro. Tiene una fuerte relación con el timbre del sonido [13].

- Cromagrama: Espectrograma en el que se organiza las frecuencias en notas musicales de la escala cromática.

Para el desarrollo y entrenamiento de los modelos se ha decidido optar por utilizar como entrada espectrogramas de Mel, que combinan los conceptos ya mencionados de espectrograma y coeficientes de Mel. Un espectrograma de Mel se trata de un espectrograma en el cual las frecuencias se han convertido a escala de Mel [14] (véase Figura 2) La escala de Mel fue desarrollada para escalar la información de frecuencias de una manera que se asemejase más a la manera en la que el ser humano percibe el sonido. Esta escala se divide en mels y cuenta con una frecuencia de referencia de 1000 kHz. Según el nivel de frecuencia, el número de mels entre unos tonos y otros que a priori están separados por la misma distancia, puede ser mayor o menor [15].

Al representar los sonidos de manera más parecida a la percepción humana, los espectrogramas de Mel son una herramienta muy útil para problemas de clasificación de audios, ya que se asemejará más al proceso que sigue el humano para este tipo de clasificaciones. Al realizar una transformada discreta de coseno (DCT) se obtienen los mencionados Mel- Frequency Cepstral Coefficients (MFCC).

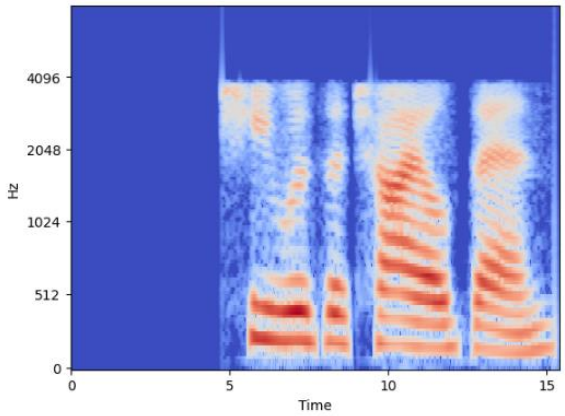

Figura 2. Espectrograma de Mel

2.4.2. Modelos de clasificación con redes neuronales convolucionales

Para la elaboración de modelos para clasificar voces reales y voces generadas por ordenador se hará uso de redes neuronales convolucionales. Este tipo de redes son idóneas para este tipo de problemas ya que se va hacer uso de imágenes a partir de audio como entrada, y este tipo de modelos está especializado en el tratamiento y clasificación de imágenes. Se trata de un algoritmo de Deep Learning que al recoger los datos de entrada, intenta detectar patrones y extraer características para poder clasificarlos en clases (véase Figura 3).

Habitualmente las redes neuronales convolucionales cuentan con tres tipos de capas:

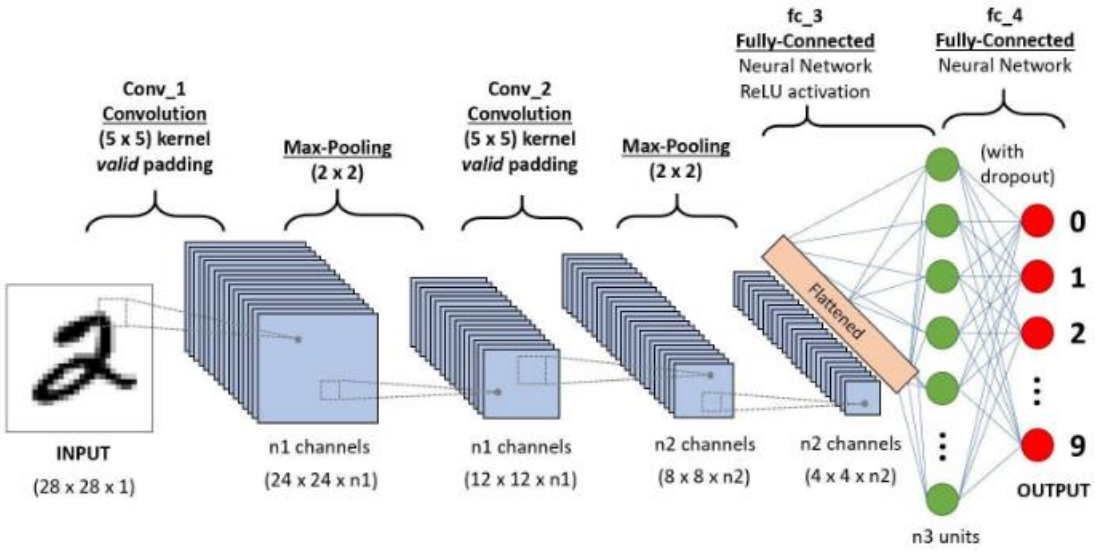

Figura 3. Ejemplo de una red neuronal convolucional [15]

Por una parte, se encuentra la red convolucional, la cual representa la base fundamental de la red. Esta capa realizará un producto matricial entre la matriz compuesta por parámetros a aprender o kernel y la sección de imagen original correspondiente a analizar. Esta operación se realiza conservando el número de canales empleados como puede ser 3 en el caso de una imagen RGB [16]. Esto produce un mapa de activación de dos dimensiones en el cual se representan los parámetros extraídos por parte del kernel en cada sección recorrida de la imagen.

Otra capa que se suele emplear en este tipo de redes es la "pooling layer", la cual se encarga de realizar una simplificación estadística de partes de la salida de la capa convolucional. Esto reduce la dimensionalidad de la representación y reduce el requerimiento computacional y los pesos a entrenar (véase Figura 4) .

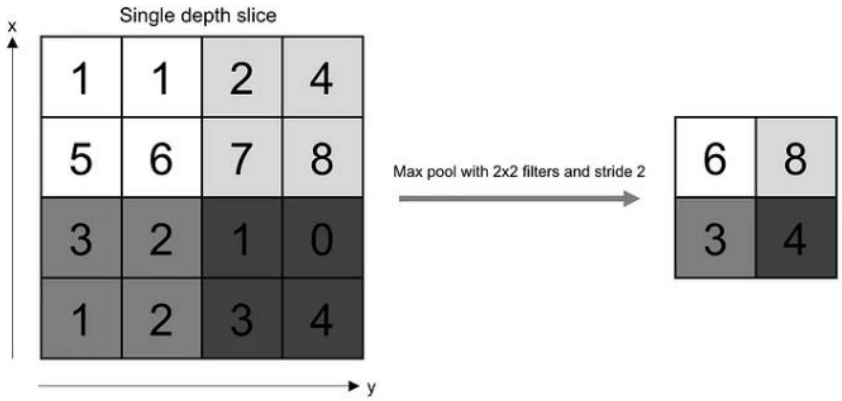

Figura 4. Operación de la capa Pooling [15]

La última capa utilizada en las redes neuronales convolucionales está formada por redes plenamente conectadas. Estas forman las capas comunes con las que cuentan todas las redes neuronales y llevan a cabo el entrenamiento de los pesos correspondientes al modelo para su ajuste [17]. Además, realizan la clasificación sugerida y mediante repetidas iteraciones y métricas de precisión se entrena el modelo para poder ofrecer los mejores resultados posibles al problema.

2.4.3. Conjunto de datos: ASVSPOOF 2021

El conjunto de datos que se utilizará para el entrenamiento de los modelos será aquel proporcionado en el reto de 2021 conocido como ASVspoof. El ASVspooof challenge (Anti-Spoofing Automatic Speaker Verification) es una competición internacional enfocada en la detección y clasificación de suplantación de identidad en sistemas de verificación del hablante. Esta competición, siendo la edición de 2021 la 4ª, se centra en la detección de este tipo de ataques, en las que se utiliza una voz generada por ordenador para hacerse pasar por un usuario real [18].

El objetivo principal del reto es incentivar la investigación y el estudio de técnicas que permitan detectar este tipo de suplantaciones para mejorar la seguridad de los sistemas de verificación de voz. Aquellos que compiten en el desafío deben diseñar algoritmos y modelos que distingan voces auténticas de voces sintéticas, siendo capaces de detectar los ataques de suplantación que tratan de hacerse pasar por un usuario mediante una voz que no es real.

El reto ofrece un conjunto de datos compuesto tanto por grabaciones de voz reales como por distintos ataques de suplantación como pueden ser voces generadas por ordenador, reproducciones de voz o conversaciones telefónicas. El ASVspoof 2021 cuenta con tres conjuntos de datos distintos:

- Acceso lógico: compuesto por grabaciones reales y falsas utilizando sistemas text-to-speech y convertidores de voz.

- Acceso físico: compuesto por grabaciones reales y grabaciones falsas consistentes en la reproducción de dichas grabaciones reales en el mismo espacio físico utilizando dispositivos de reproducción.

- Speech Deepfake: compuesto por grabaciones reales y falsas utilizando sistemas text-to-speech y convertidores de voz. Similar al conjunto de acceso lógico, pero sin contar con verificación del hablante.

Para la elaboración de los modelos se utilizará únicamente el conjunto de datos de acceso lógico, ya que es el que cuenta con más información y tiene más variantes de ataques de suplantación. Este conjunto de datos tiene un espacio de 7.8 GB. Todos los archivos de audio cuentan con formato .flac y están muestreados a 16 kHz.

Junto a los archivos de audio, se proporciona un fichero con información acerca de los distintos campos de las grabaciones. Estos son:

- Identificador del hablante: LA_XXXX.
- Nombre del archivo: LA_E_XXXXXXX.
- Tipo de códec.
- Tipo de ataque: AXX.
- Clase: bonafide (real) o spoof (artificial).
- Trim: si está recortado (trim) o no (notrim) el audio.

2.5. Herramientas

2.5.1. Resemble.AI

Para realizar las pruebas de seguridad en los asistentes personales se necesitará un sistema de clonación de voces generadas por inteligencia artificial. Con esta aplicación se recreará un modelo basado en la voz del autor del presente TFG en este caso y se intentarán realizar suplantaciones de identidad en una serie de dispositivos mediante la reproducción de varios comandos de activación. En este caso se ha optado por utilizar Resemble.AI como software para la clonación de voz.

Resemble.AI es una página web de uso público que permite clonar voces fácilmente y después reproducir lo que se quiera con sistemas text-to-speech. En primer lugar, se requiere el envío de un archivo de audio de una voz de al menos 3 minutos de duración o la lectura de 25 frases que proporciona la aplicación. Una vez recibidos los datos, el sistema tarda unos 12 minutos en crear un modelo basado en la voz proporcionada el cual se podrá utilizar para lo que pretenda el usuario, ya sea para narraciones, actuaciones o experimentación como es el caso de este proyecto.

El equipo de Resemble.AI está compuesto por una serie de ingenieros, especializados en Deep Learning en su mayoría, originalmente establecidos en Toronto y ahora distribuyéndose alrededor del mundo en lugares como San Francisco, Los Ángeles o España. La plataforma está viviendo una evolución muy rápida en el último año y además de refinar el actual sistema de clonación de voces del que disponen, están ofreciendo nuevas herramientas como son sistemas de clonación de voz a tiempo real, uso de determinadas emociones y gestión de pausas a la hora de reproducir textos y expansión a distintos idiomas ya que originalmente se limitaba únicamente al inglés. La página cuenta con una versión de pago, la cual da acceso a mayor cantidad de voces para

almacenar y otorga acceso a sistemas que aún están en prueba, pero para el desarrollo de este trabajo se contará con la versión gratuita que será suficiente para realizar los modelos de voz requeridos para los ataques de suplantación de identidad [19].

2.5.2. librosa

librosa se trata de una librería de Python especializada en análisis de música y audio. Permite extraer características de ficheros de audio tales como espectrogramas, separación de frecuencias y decodificación de audio [20].

2.5.3. Tensorflow

Tensorflow es una librería de Python ampliamente utilizada a la hora de desarrollar modelos de aprendizaje automático ya sean previamente entrenados o nuevos modelos. Se trata de una librería de código abierto desarrollada por Google que permite desarrollar todo tipo de redes neuronales y posteriormente entrenarlas a partir de la detección de patrones y características de los datos introducidos [21].

2.5.4. Skicit Learn

Skicit Learn o SKlearn es una librería de Python que otorga acceso a una serie de funcionalidades y herramientas relacionadas con Machine Learning. Incluye algoritmos de clasificación y regresión entre otros, así como métricas de precisión y reducción de dimensionalidad. Tiene una alta compatibilidad con otras librerías como NumPy o matplotlib [22].

Capítulo 3. Pruebas de Seguridad con Smart Personal Assistants

En este capítulo se realizarán una serie de pruebas para comprobar la seguridad y protección de datos de algunos de los asistentes personales más utilizados como son Siri, por parte de Apple, Alexa por parte de Amazon y Google Assistant por parte de Google.

A continuación, se hará una descripción de las pruebas a realizar para cada asistente personal, las cuales se enfocarán en cómo de seguro es el dispositivo a la hora de enfrentarse a distintas posibilidades de acceder a estos sistemas. Además, se analizará qué aplicaciones pueden estar comprometidas en caso de que se dé un acceso de los previamente mencionados y cómo de protegidos están los datos del usuario en cada dispositivo en el caso de que se produzcan estos ataques.

3.1. Descripción de las pruebas

Por una parte, las pruebas consisten en realizar una serie de intentos de suplantación en los dispositivos mediante distintos tipos de ataques para acceder al dispositivo. Por otra parte, se especificará según el dispositivo, en caso de que el atacante pueda acceder a este, qué aplicaciones con información comprometida pueden verse afectadas y con qué capas extra de seguridad cuentan en caso de hacerlo. Por lo tanto, las pruebas se dividen en dos secciones:

- Ataques de suplantación de identidad por voz. Se tratará de acceder al dispositivo y traspasar la seguridad del asistente personal con distintos métodos.
    - Grabación de voz. Se reproducirá una muestra de la voz original del usuario pronunciando determinados comandos.
    - Voz Text-to-Speech genérica. Se reproducirá una voz generada por ordenador mediante el sistema text-to-speech ofrecido por Google Cloud [23]. La voz empleada será una voz predeterminada ofrecida por la plataforma.

- Voz generada por inteligencia artificial. Mediante el uso de la aplicación Resemble.AI, se utilizará un modelo de voz artificial basado en la voz del usuario del dispositivo con el que se reproducirán los comandos necesarios para acceder al dispositivo en cuestión.

- Acceso a aplicaciones comprometidas o con información privada

    - Aplicaciones de banca y finanzas.
    - Aplicaciones de compras on-line.
    - Aplicaciones de mensajería.
    - Calendarios y tareas.

Antes de explicar el desarrollo de las pruebas como tal han de realizarse una serie de apreciaciones. En primer lugar, los asistentes personales han sido puestos a punto en base a la voz del autor del presente proyecto, por lo que todos contarán con el mismo individuo como usuario a verificar. Del mismo modo, la recreación de la voz hecha por el programa Resemble.AI fue entrenada gracias a audios y grabaciones de la misma persona.

Por otra parte, cabe mencionar que, debido al uso de este software de clonación de voz, esta se ha realizado en inglés, ya que era el único idioma permitido para realizar la recreación. Por tanto, a su vez, todos los dispositivos han sido configurados en este mismo idioma y los comandos de activación y la comunicación en general con los dispositivos se llevará a cabo en esta lengua.

3.2. Siri (iPhone 11)

Siri es un asistente personal virtual desarrollado por Apple presentado en 2011 por primera vez al mismo tiempo que el lanzamiento del iPhone 4S. Gracias a tecnologías de procesamiento de lenguaje natural y reconocimiento de voz permite analizar y responder cuestiones y comandos realizadas por el usuario.

Siri se ha diseñado de tal manera que permite una interacción en los dispositivos en los que se encuentra para acceder de una manera rápida y simple a aplicaciones como Mail, Contactos, Mensajes, Mapas y muchas más [24].

Una diferencia con la que cuenta Siri en comparación con el resto de asistentes personales es la integración con desarrollo de terceras partes. Alexa, por ejemplo, es un asistente que permite mucha más funcionalidad de terceros por medio de sus "skills" al igual que hace Google Assistant con las "actions". En cambio, Siri cuenta con funcionalidades mucho más restringidas y todas ellas supervisadas previamente por Apple.

Actualmente Siri se encuentra disponible en todo tipo de dispositivos Apple como son iPhone, iPad, AirPods, Apple Watch, HomePod, Mac, Apple TV y Carplay [25]. Para el desarrollo de las pruebas de este trabajo se ha optado por hacer uso de Siri en un dispositivo iPhone 11 con el sistema operativo IOS 15.6.1 (véase Figura 5).

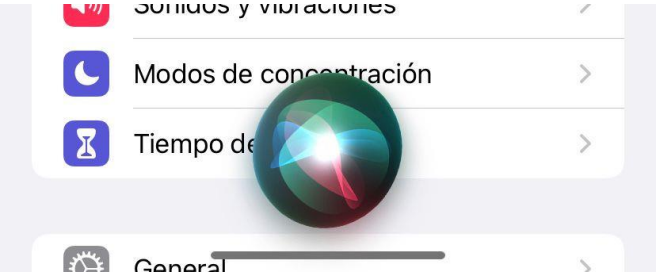

Figura 5. Interfaz de uso de Siri en iPhone 11

3.2.1. Ataques de suplantación de identidad por voz

En el caso de Siri, para activar el asistente personal se ha de pronunciar el comando "Hey Siri"y una vez se escuche este, el asistente estará en un modo de escucha para responder a las consultas que se realicen. La configuración de Siri permite configurar el asistente en caso de que se desee para que se active al escuchar este comando y sólo reconozca al usuario que lo ha configurado. En caso de que no se quiera contar con esto, se puede eliminar la escucha de este comando y Siri únicamente funcionará cuando se mantenga el botón de encendido del dispositivo y podrá ser utilizado por cualquiera (véase Figura 6) . Para las pruebas a realizar se mantendrá la escucha del comando.

Figura 6. Pantalla de configuración de Siri

Es importante mencionar que, en el caso de Siri, una vez se pronuncia la frase de activación y esta es reconocida por el dispositivo, el asistente aceptará comandos y consultas de cualquier voz, sin ser necesariamente la del usuario original. Por lo tanto, este primer reconocimiento de voz en el comando de activación será la única barrera de cara a poder acceder o no al contenido del dispositivo.

Los resultados de las pruebas realizadas se muestran a continuación y, en versión resumida, en la Tabla 1.

- Ataque con grabación de voz: Reproduciendo una grabación del usuario original pronunciando el comando de activación "Hey Siri", el asistente **reconoce como real** dicha voz, permitiendo acceder al resto del contenido.

- Ataque de voz text-to-speech: Reproduciendo el comando de activación "Hey Siri" mediante una herramienta de text-to-speech, el asistente **no reconoce como real** dicha voz, no permitiendo que el atacante pueda acceder al dispositivo

- Ataque de voz generada por inteligencia artificial (Resemble.AI). La reproducción del comando de activación "Hey Siri" a través de la recreación de la voz del usuario generada por inteligencia artificial (véase Figura 7) **se reconoce como real**, permitiendo acceder al resto del contenido.

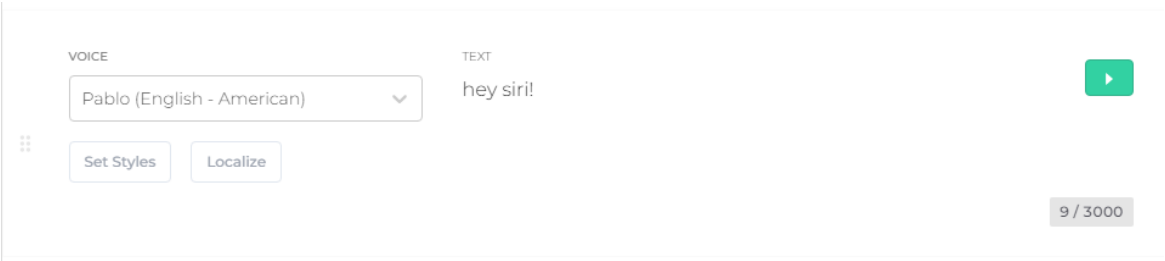

Figura 7. Reproducción del comando "Hey Siri" mediante voz generada por IA

Tabla 1. Resumen de resultados de ataques de suplantación en Siri

Método de suplantación de voz	**Resultado**
Grabación de voz	Sí es reconocida como la voz original
Voz text-to-speech	No es reconocida
Voz generada por inteligencia artificial	Sí es reconocida como la voz original

3.2.2. Acceso a aplicaciones comprometidas

A la hora de analizar cómo se comportan los dispositivos en lo referente a permitir acceder a aplicaciones determinadas, se tratará de hacer uso de estas inicialmente desde el asistente y se observará qué tipo de seguridad se presenta a medida que se vayan realizando consultas respecto a la aplicación en cuestión. Estos intentos de acceso se realizarán bajo la premisa de que el asistente reconoce como real la voz utilizada, y por tanto otorga acceso al uso del dispositivo. También cabe la posibilidad de que el asistente no cuente con la capacidad de ofrecer acceso a este tipo de aplicaciones en cuyo caso se anotará, indicando qué aplicación se ha utilizado. Es posible que aplicaciones que sí cuentan con funcionalidades relacionadas con los asistentes en otros países, vean restringidos estos permisos en España o viceversa.

Los resultados se muestran a continuación y, en versión resumida, en la Tabla 2.

- Aplicaciones de banca y finanzas: En este caso se ha utilizado la aplicación del banco BBVA para intentar realizar transacciones desde el asistente personal. A pesar de que sí ha permitido realizar consultas relacionadas con la aplicación, una vez se querían realizar operaciones, se requería desbloquear el teléfono con la contraseña en cuestión. Una vez se inciaba la aplicación, esta requería de las credenciales correspondientes para acceder a ella.

- Aplicaciones de compras on-line: Siri no permite realizar compras on-line en sus dispositivos debido a la política de privacidad de Apple [26].

- Aplicaciones de mensajería: Siri permite enviar mensajes y correos electrónicos a través de Siri únicamente, sin necesidad además de desbloquear el teléfono. Se ha probado esta funcionalidad en aplicaciones como WhatsApp y Gmail (véase Figura 8).

- Calendarios y tareas: Requiere de desbloqueo del teléfono para acceder a la información.

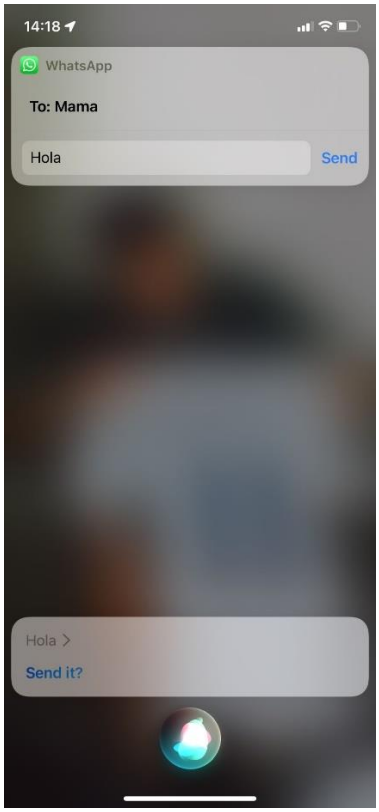

Figura 8. Ejemplo de un envío de mensaje WhatsApp a través de Siri

Tabla 2. Resumen de resultados de pruebas de acceso a aplicaciones comprometidas en Siri

Tipo de aplicación	**Resultado**
Banca y Finanzas (BBVA)	Requiere de desbloqueo de teléfono y credenciales de la aplicación
Compras on-line	Esta funcionalidad no está permitida desde Siri
Mensajería (WhatsApp)	Permite enviar mensajes y correos electrónicos desde la pantalla de bloqueo
Calendarios y tareas (Calendar, Google Calendar)	Requiere de desbloqueo de teléfono

3.3. Alexa (Amazon Echo Show 5)

Alexa es un asistente virtual desarrollado por Amazon, introducido al mercado por primera vez en 2014 a la vez que el dispositivo Amazon Echo. Al igual que Siri, utiliza tecnología de reconocimiento de voz y procesado de lenguaje natural para comunicarse con el usuario y responder a las consultas deseadas. En un primer momento Alexa únicamente funcionaba con los dispositivos de Amazon hasta que la empresa decidió permitir que otras empresas pudieran construir sus dispositivos en torno al asistente. Esto provocó una expansión de dispositivos conectados a Alexa, desde cerraduras y lámparas hasta microondas [27].

Además de la integración con terceros, Alexa cuenta con las denominadas skills, que serían el equivalente a aplicaciones desarrolladas para dichos dispositivos, las cuales permiten ofrecer funcionalidades adicionales a las predeterminadas. Alexa se encuentra disponible en dispositivos como altavoces inteligentes (línea Echo Pop y Echo Dot), pantallas inteligentes (línea Echo Show), televisiones o extensiones de los mismos (Amazon Fire) u otros dispositivos como enchufes y lámparas (Amazon Smart Plug) [28].

En el caso de este proyecto, se utilizará el modelo Amazon Echo Show 5 de 1ª generación, el cual salió a la venta en 2019 Además de los habituales controles de volumen y encendido, cuenta con pantalla táctil, así como cámara integrada [Figura 9]. Para reforzar la seguridad, el dispositivo permite desactivar el micrófono y tapar la cámara físicamente.

Figura 9. Amazon Echo Show 5

3.3.1. Ataques de suplantación de identidad por voz

En el caso de Alexa, el asistente permite acceder al dispositivo con el comando "Alexa, {consulta a realizar}". Este comando puede ser pronunciado por cualquier persona y el dispositivo se activará y responderá a la petición. A pesar de ello, como capa adicional de seguridad, desde la configuración del dispositivo se pueden crear perfiles de personas asociadas a determinadas voces (véase Figura 10) . De esta manera, cuando una persona con un perfil creado se comunique con Alexa, esta será reconocida y tendrá acceso a determinadas aplicaciones y privilegios con los que no podría contar si no fuese reconocido por el dispositivo. Será este último caso en el cual se basarán las pruebas a continuación para verificar los sistemas de seguridad y autenticación del dispositivo.

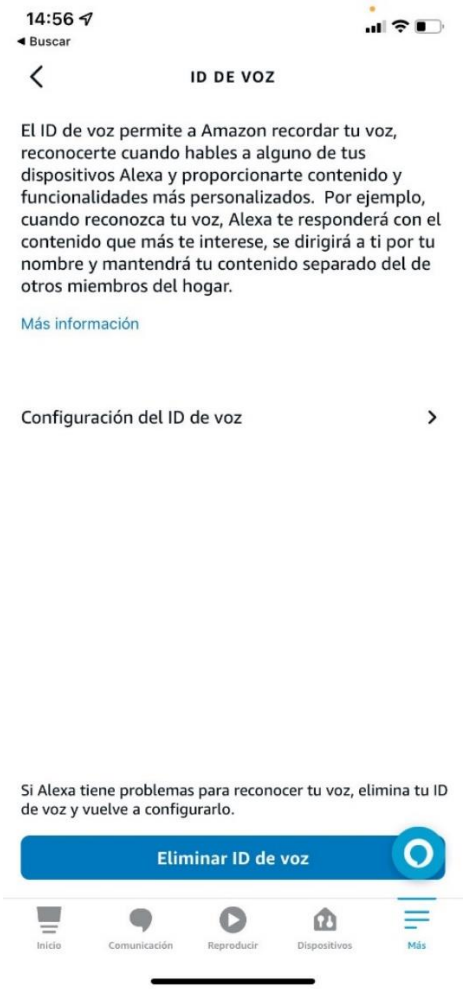

Figura 10. Pantalla de configuración de perfil en Alexa desde dispositivo móvil

Los resultados de las pruebas realizadas se muestran a continuación y, en versión resumida, en la Tabla 3.

- Ataque con grabación de voz: Reproduciendo una grabación del usuario original pronunciando el comando de activación "Alexa, {orden}", el asistente **reconoce como real** dicha voz, permitiendo acceder al dispositivo siendo identificado como usuario perteneciente al perfil creado.

- Ataque de voz text-to-speech: Reproduciendo el comando de activación "Alexa, {orden}" mediante una herramienta de text-to-speech, el asistente **no reconoce como real** dicha voz, permitiendo acceder al contenido básico del dispositivo, pero sin ser identificado como usuario de un perfil.

- Ataque de voz generada por inteligencia artificial (Resemble.AI). La reproducción del comando de activación "Alexa, {orden}" a través de la recreación de la voz del usuario generada por inteligencia artificial (véase Figura 11) **se reconoce como real**, permitiendo acceder al dispositivo siendo identificado como usuario perteneciente al perfil creado.

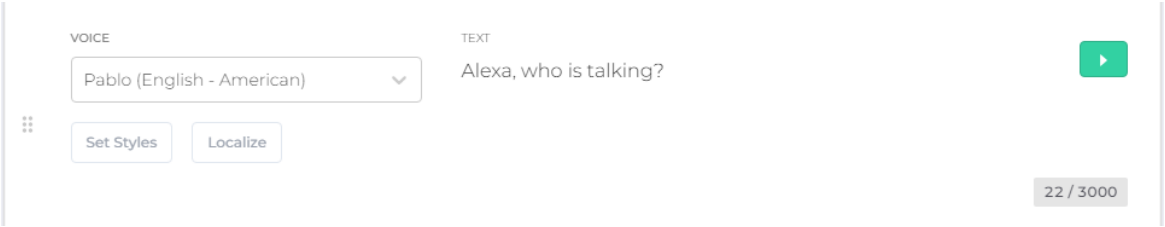

Figura 11. Reproducción del comando "Alexa, who is talking?" mediante voz generada por IA

Tabla 3. Resumen de resultados de ataques de suplantación en Alexa

Método de suplantación de voz	**Resultado**
Grabación de voz	Sí es reconocida como voz de perfil creado
Voz text-to-speech	No es reconocida como voz de perfil
Voz generada por inteligencia artificial	Sí es reconocida como voz de perfil creado

3.3.2. Acceso a aplicaciones comprometidas

Para valorar el acceso a aplicaciones comprometidas utilizando Alexa, se tratará de ver si se pueden utilizar las funcionalidades a comentar autenticando una persona en el dispositivo como poseedor de un perfil para contar con todos los privilegios posibles a la hora de utilizar el dispositivo.

Los resultados de los intentos de acceso se muestran a continuación y, en versión resumida, en la Tabla 4.

- Aplicaciones de banca y finanzas: En el caso de Alexa, no se ha podido comprobar que permita acceder a datos bancarios o realizar operaciones ya que las aplicaciones de las entidades consultadas (BBVA, Santander) son meramente informativas y no permite acceder a cuentas ni realizar transacciones.

- Aplicaciones de compras on-line: Mediante el uso de Alexa, se pueden realizar a través de ella compras en la tienda de Amazon. De manera predeterminada se requiere de un perfil para realizar las compras aunque este bloqueo puede sustituirse por un código de voz o puede eliminarse por completo. (véase Figura 12)

Figura 12. Pantalla de configuración de compra por voz en Alexa

- Aplicaciones de mensajería: Alexa únicamente permite enviar mensajes a otros usuarios con Alexa que se hayan guardado previamente en los contactos o se hayan sincronizado con aquellos de otro dispositivo.

- Calendarios y tareas: Permite ver eventos y realizar modificaciones en el calendario sin necesidad de identificar el perfil.

Tabla 4. Resumen de resultados de pruebas de acceso a aplicaciones comprometidas en Alexa

Tipo de aplicación	**Resultado**
Banca y Finanzas (BBVA, Santander)	Aplicaciones informativas. No hay opción a acceder a cuentas o realizar transacciones
Compras on-line (Amazon)	Posibilidad de realizar compras mediante identificación de perfil (configuración predeterminada)
Mensajería (Alexa)	Permite enviar mensajes a otros usuarios de Alexa
Calendarios y tareas (Google Calendar)	Permite acceder a la información de eventos y realizar modificaciones sin verificar el perfil

3.4. Google Assistant (Google Home Mini)

Google Assistant es un asistente virtual creado por Google que emplea inteligencia artificial para interactuar con usuarios y ofrecerles información y ayuda en diversas tareas. Puede ser utilizado mediante dispositivos móviles, altavoces inteligentes, relojes inteligentes y otros dispositivos que sean compatibles [29].

El modelo que se ha utilizado para las pruebas es el altavoz inteligente Google Home Mini lanzado en Octubre de 2017 (véase Figura 13) . El dispositivo únicamente cuenta con un botón físico para activar o desactivar el micrófono, dejando el resto de controles como volumen y encendido y apagado a botones táctiles.

Figura 13. Google Home Mini

3.4.1. Ataques de suplantación de identidad por voz

En el caso de Google Asisstant, el dispositivo se activará cuando escuche los comandos "Hey Google" o "OK Google". El asistente de Google permite acceder al dispositivo a cualquier persona, pero de manera similar a Alexa, reconoce la persona que lo está utilizando en caso de que se haya guardado. Esto restringirá el acceso a determinadas funcionalidades como calendarios y eventos.

Los resultados de las pruebas realizadas se muestran a continuación y, en versión resumida, en la Tabla 5.

- Ataque con grabación de voz: Reproduciendo una grabación del usuario original pronunciando el comando de activación "Hey Google" o "OK Google", el asistente **reconoce como real** dicha voz, permitiendo acceder al dispositivo y siendo identificado.

- Ataque de voz text-to-speech: Reproduciendo el comando de activación "Hey Google" o "OK Google" mediante una herramienta de text-to-speech, el asistente **no reconoce como real** dicha voz, permitiendo acceder al contenido del dispositivo pero sin identificar a la persona

- Ataque de voz generada por inteligencia artificial (Resemble.AI): Reproduciendo una grabación del usuario original pronunciado el comando de activación "Hey Google" o "OK Google" (véase Figura 14) , el asistente **reconoce como real** dicha voz, permitiendo acceder al dispositivo y siendo identificado.

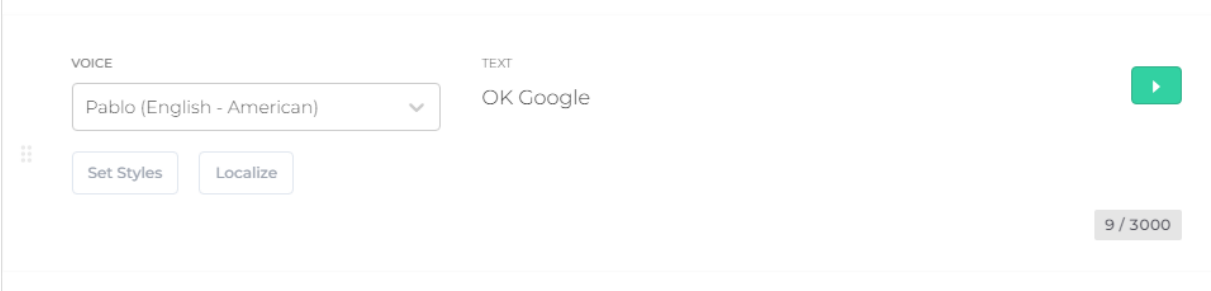

Figura 14. Reproducción del comando "OK Google" mediante voz generada por IA

Tabla 5. Resumen de resultados de ataques de suplantación en Google Assistant

Método de suplantación de voz	**Resultado**
Grabación de voz	Sí es reconocida como persona conocida
Voz text-to-speech	No es reconocida como voz de perfil
Voz generada por inteligencia artificial	Sí es reconocida como persona conocida

3.4.2. Acceso a aplicaciones comprometidas

Para valorar el acceso a aplicaciones comprometidas utilizando el asistente de Google, se tratará de ver si se pueden utilizar las funcionalidades a comentar autenticando un perfil en el dispositivo para contar con todos los privilegios posibles a la hora de utilizar el dispositivo.

Los resultados se muestran a continuación y, en versión resumida, en la Tabla 6.

- Aplicaciones de banca y finanzas: No existen aplicaciones de este tipo.

- Aplicaciones de compras on-line: Las compras on-line desde Google Assistant están restringidas únicamente a Estados Unidos.

- Aplicaciones de mensajería: No se pueden escribir mensajes ni realizar llamadas mediante Google Assistant.

- Calendarios y tareas: Permite acceso a información acerca de calendarios y eventos únicamente si se reconoce a la persona.

Tabla 6. Resumen de resultados de pruebas de acceso a aplicaciones comprometidas en Alexa

Tipo de aplicación	**Resultado**
Banca y Finanzas	No se dispone de esta funcionalidad
Compras on-line	Funcionalidad restringida a Estados Unidos
Mensajería	No se dispone de esta funcionalidad
Calendarios y tareas (Google Calendar)	Permite acceder a la información de eventos y realizar modificaciones tras identificación previa

Capítulo 4. Desarrollo de modelos de Deep Learning para clasificación de voces reales y voces sintéticas

Para este apartado se diferenciarán 2 capítulos distintos. Por una parte, se explicará el tratamiento y preprocesado de los datos para su posterior uso. Por otro lado, se explicará el desarrollo de los modelos que se han utilizado junto a sus procesos de entrenamiento y validación.

4.1. Preprocesado de datos

4.1.1. Extracción de datos

En primer lugar se extraen los datos del conjunto de datos ASVspoof 2021 que se ha mencionado en la sección previa de estado del arte. Para facilitar el uso de esos registros, se han eliminado campos que resultaban irrelevantes para el problema como son el tipo de códec, el tipo de ataque de suplantación o si el audio está recortado. Además, se ha añadido un campo con la ruta del archivo de audio en cuestión para poder acceder al mismo de manera más cómoda en los siguientes pasos.

Debido a que el dataframe original ofrecido por ASVspoof cuenta con más de 180.000 registros y esta cantidad de datos supondría una potencia de cómputo y tiempo de procesamiento muy elevados, se seleccionará una muestra de 1000 registros, los cuales se compondrán de muestras de 500 registros aleatorios correspondientes a voces reales y otros 500 de voces sintéticas.

Además, esta muestra total de 1000 registros se dividirá en una muestra para el entrenamiento del modelo (75%) y una muestra como test (25%) para comprobar los niveles de predicción del modelo con valores nuevos. Por otro lado, se separarán en estas muestras la clase a la que pertenece el audio con el resto de la información.

4.1.2. Procesado de audios

De cara a ofrecer al modelo como entrada audios uniformes y variar los audios originales para que no se produzcan sesgos, se realizarán una serie de manipulaciones a los audios del dataset. Por una parte de hará un recorte para eliminar silencios iniciales y finales del audio, se ajustará el mismo a una duración determinada y por último se moverá aleatoriamente el audio en el eje temporal (Time Shifting) (véase Figura 15 y Figura 16) .

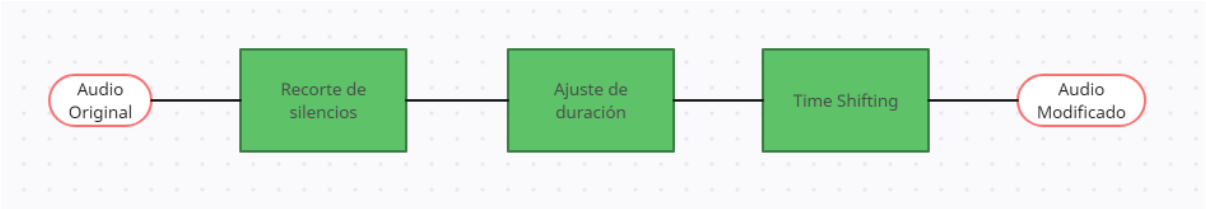

Figura 15. Diagrama de modificación de archivo de audio

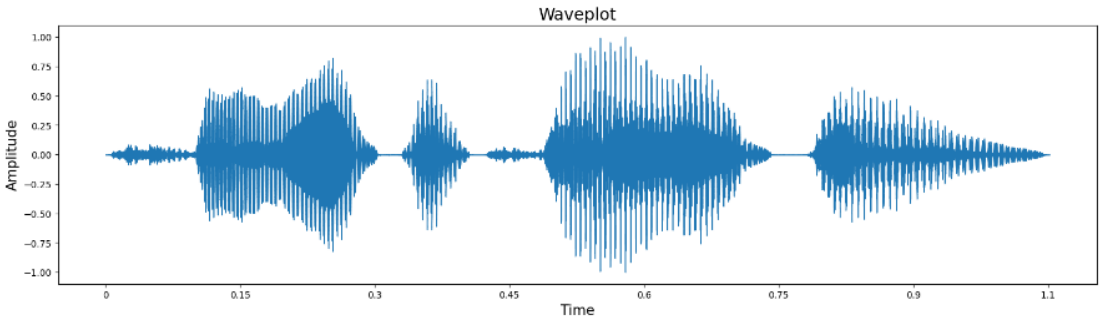

Figura 16. Ejemplo de forma de onda de audio original

En primer lugar, se realizará el recorte de silencios iniciales y finales en el archivo de audio. Para este proceso de modificación del audio, se utilizarán funciones basadas en el código ofrecido por el usuario de Kaggle "Awsaf" en su desarrollo del reto de ASVspoof propuesto en 2019 [30]. La función utilizada para el recorte de audio [Código 1] y el resultado de este [Figura 17] se muestran a continuación.

```
# Trim
def TrimAudio(audio, epsilon=0.15):
    pos  = tfio.audio.trim(audio, axis=0, epsilon=epsilon)
    audio = audio[pos[0]:pos[1]]
    return audio
```

Código 1. Función para recorte de audio

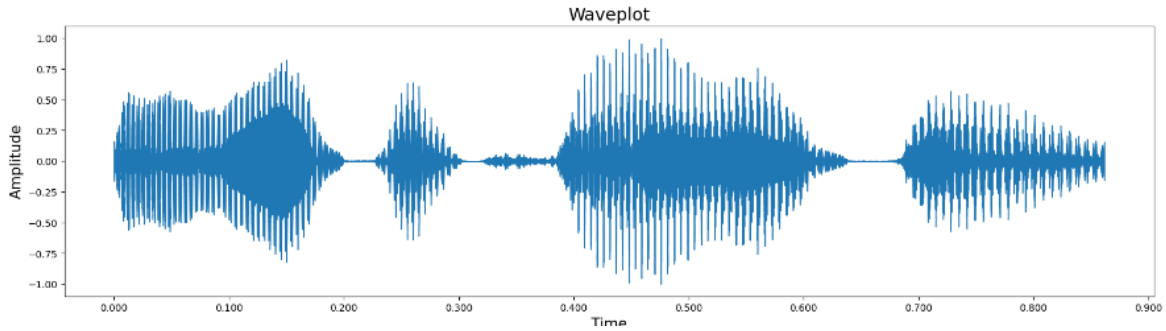

Figura 17. Ejemplo de forma de onda de audio sin silencios iniciales y finales

A continuación, se realiza un ajuste de duración en los audios para mantener la misma duración en todos los archivos. En el desarrollo de este modelo se ha elegido una duración de 1,5 segundos ya que se trata de una duración en la que suficientes archivos pueden aportar información y además no supone un almacenamiento computacional demasiado elevado. La función empleada prolongará con ceros al principio y al final del audio en caso de que sea demasiado corto, o lo recortará en caso de que exceda esta duración.

La función empleada [Código 2] y el resultado correspondiente [Figura 18] se muestran a continuación.

```
# Crop or Pad audio to keep a fixed length
def CropOrPad(audio, target_len, pad_mode='constant'):
    audio_len = tf.shape(audio)[0]
    if audio_len < target_len: # if audio_len is smaller than target_len then use
Padding
        diff_len = (target_len - audio_len)
        pad1 = random_int([], minval=0, maxval=diff_len) # select random location for
padding
        pad2 = diff_len - pad1
```

```
        pad_len = [pad1, pad2]
        audio = tf.pad(audio, paddings=[pad_len], mode=pad_mode) # apply padding
    elif audio_len > target_len:  # if audio_len is larger than target_len then use
Cropping
        diff_len = (audio_len - target_len)
        idx = tf.random.uniform([], 0, diff_len, dtype=tf.int32) # select random
location for cropping
        audio = audio[idx: (idx + target_len)]
    audio = tf.reshape(audio, [target_len])
    return audio
```

Código 2. Función para ajuste de duración de audio

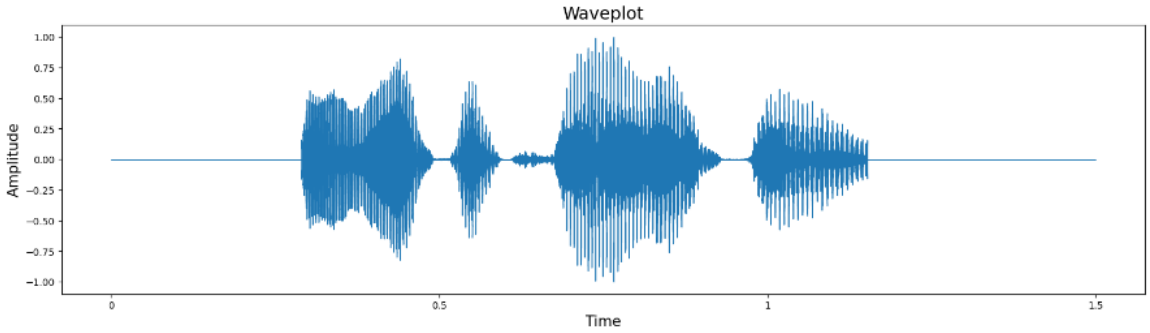

Figura 18. Ejemplo de forma de onda de audio sin silencios iniciales y finales y ajustado a 1,5 segundos

Por último, se realiza una variación aleatoria en el eje temporal del audio para mezclar los datos del mismo y emplear un audio nuevo generado en base al audio original. La función empleada [Código 3] y el resultado [Figura 19] se muestran a continuación.

```
# Tools
def random_int(shape=[], minval=0, maxval=1):
    return tf.random.uniform(shape=shape, minval=minval, maxval=maxval,
dtype=tf.int32)


def random_float(shape=[], minval=0.0, maxval=1.0):
    rnd = tf.random.uniform(shape=shape, minval=minval, maxval=maxval,
dtype=tf.float32)
    return rnd


# Randomly shift audio -> any sound at <t> time may get shifted to <t+shift> time
def TimeShift(audio, prob=0.5):
    if random_float() < prob:
        shift = random_int(shape=[], minval=0, maxval=tf.shape(audio)[0])
        if random_float() < 0.5:
            shift = -shift
        audio = tf.roll(audio, shift, axis=0)
    return audio
```

Código 3. Función para modificación temporal aleatoria de audio

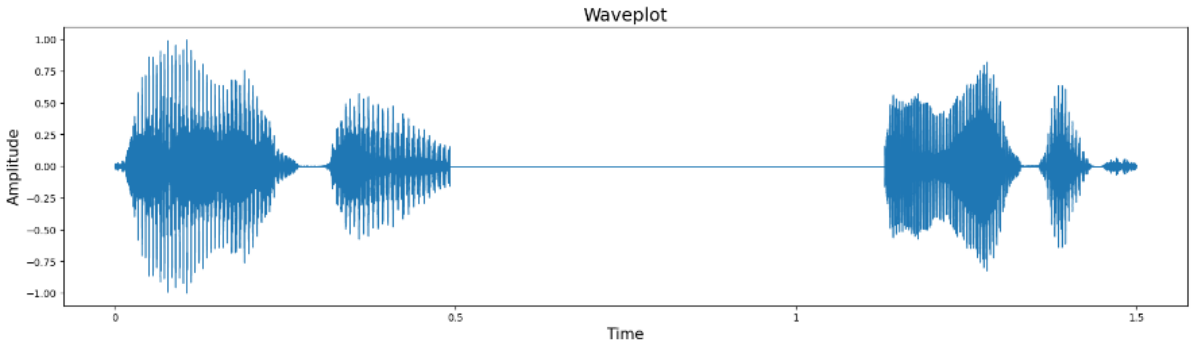

Figura 19. Ejemplo de forma de onda de audio generado aleatoriamente en base a audio sin silencios iniciales y finales y ajustado a 1,5 segundos

4.1.3. Generación de datos de entrada (espectrogramas de Mel) en el modelo

Como se ha comentado en secciones previas, se emplearán espectrogramas de Mel extraídos a partir de los archivos de audio como dato de entrada para el entrenamiento del modelo. Estos espectrogramas se representarán en escala logarítmica para poder tener una visualización más clara del mismo.

Para poder ajustar el espectrograma y los archivos de audio previos de manera que los datos estuviesen representados correctamente y se pudiesen extraer características de ellos se han configurado una serie de parámetros tanto para el audio como para los espectrogramas, los cuales son los siguientes:

- sample_rate: número de muestras que se toman por segundo. En este caso se ha escogido un valor de **16 kHz** ya que es la frecuencia a la que se han muestreado originalmente los audios.

- audio_len: número de muestras totales en el audio. Al estar todos los audios ajustados a 1,5 segundos, el valor será de **24.000 muestras**.

- n_fft: longitud de segmentos a los que se aplica la transformada rápida de Fourier (FFT) durante la generación del espectrograma. Debido a la longitud del audio se ha optado por un valor de **512 muestras**.

- hop_length: distancia de salto entre dos segmentos. Se utilizará un valor de **50 muestras**.

Para generar los espectrogramas de Mel de todos los registros de la muestra de 1000 audios, se ha desarrollado una función que permitirá realizarlo teniendo como entrada simplemente el dataframe del cual se quieren extraer los espectrogramas. En este función se recorrerá el dataframe, extrayendo por cada registro la ruta en la que se encuentra el audio y la clase a la que pertenece (sintético o real). Posteriormente se carga el audio gracias a la ruta extraída, se normaliza y se le aplica el preprocesado mencionado anteriormente. Más tarde se genera el espectrograma de Mel en escala logarítmica. El espectrograma tiene una dimensión de (128,481) pero como la red neuronal convolucional espera una tercera dimensión que muestre el número de

canales de la imagen, en este caso 1, se cambiará la dimensionalidad del espectrograma a (128,481,1). Finalmente, como salidas de la función se tiene una vector con todos los espectrogramas extraídos, así como un vector con las clases correspondientes.

La función íntegra se muestra en el Código 4.

```
def generate_mel_spectrograms(df):

    # df: dataframe con el que se trabaja

    features = []
    labels = []
    df = df.reset_index()

    for i in range(len(df)):
        path = df.iloc[i]['filepath']
        class_id = df.iloc[i]['class']
        y, sr = librosa.load(path, sr=sample_rate)
        y = Normalize(y)

        print(path)

        # Preprocesado
        y = TrimAudio(y)
        y = CropOrPad(y, audio_len)
        y = TimeShift(y)
        #y = GaussianNoise(y) # Opcion a introducir ruido gaussiano
        y = y.numpy()

        # Espectrograma de Mel
        mel_spectrogram = librosa.feature.melspectrogram(y=y, sr=sr, n_fft=n_fft, hop
_length=hop_length, fmax=8000)
        log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)
        log_mel_spectrogram = np.reshape(log_mel_spectrogram, (128, 481, 1))

        features.append(log_mel_spectrogram[np.newaxis,...])
        labels.append(class_id)

    features, labels = np.concatenate(features, axis=0), np.array(labels)

    # features: array de espectrogramas extraídos
    # labels: array de clases


    return features, labels
```

Código 4. Función de generación de espectrogramas de Mel

Como último paso antes de introducir los datos al modelo de redes neuronales se ha modificado el formato de las etiquetas de las clases pasando de ser string a ser una clase binaria (0: sintética, 1: real), lo cual permite al modelo realizar una clasificación en base a este tipo de datos.

4.2. Desarrollo de modelos de clasificación

Para desarrollar los modelos de voz se ha empleado la librería Tensorflow, especialmente la API de alto nivel Keras. Se ha optado por emplear dos tipos de modelos. Por un lado, se han desarrollado dos modelos de redes neuronales convoluciones desde cero eligiendo las capas empleadas, funciones de activación y demás parámetros. Por otro lado, también se han empleado modelos pre-entrenados comunes para este tipo de problemas de clasificación, adaptándolos al caso en cuestión.

Estos modelos se han entrenado con el 85% del set de entrenamiento mencionado, dejando el 15% restante para un proceso de validación durante el entrenamiento para observar si se produce un posible sobreajuste. Posteriormente, se probarán los modelos con el set de test para comprobar su rendimiento con datos desconocidos.

4.2.1. Modelo original I (2 capas Conv2D)

Como primer modelo original a entrenar se empleará un modelo de redes neuronales convolucionales con distintos tipos de capas. Principalmente el modelo contará con 2 capas de convolución de dos dimensiones con tamaño de kernel de 3x3, 2 capas de MaxPooling de tamaño 2x2 para reducir la dimensionalidad de los datos, 2 capas de Dropout de tasa 0,4 y otra de 0,2 para reducir el sobreajuste y una capa plenamente conectada de 128 neuronas.

La función de activación será softmax para realizar la clasificación entre las 2 clases a seleccionar. El modelo utilizará de función de pérdida binary_crossentropy, comúnmente utilizada en problemas binarios de clasificación [31]. La función de optimización será Adam y el modelo se entrenará durante 15 épocas. El código empleado para el modelo es el siguiente [Código 5].

```
# Modelo 1

model1 = Sequential()

model1.add(Input(shape=(128, 481,1)))
model1.add(Conv2D(32, kernel_size=(3, 3), activation='relu',kernel_regularizer=regula
rizers.l2(l=0.01)))
model1.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model1.add(Dropout(0.4))
model1.add(Conv2D(64, kernel_size=(3, 3), activation='relu',kernel_regularizer=regula
rizers.l2(l=0.01)))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.4))
model1.add(Flatten())
model1.add(Dense(128, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Dense(2, activation='softmax'))
model1.summary()


model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model1.fit(train_features, train_labels_encoded, epochs=15, validation_spli
t=0.2)
```

Código 5. Modelo original I

Los resultados para el entrenamiento del modelo han sido de 0,76 de precisión para el entrenamiento y 0,56 para la validación.

La evolución de las precisiones y la pérdida durante las distintas épocas se muestran a continuación [Figura 20].

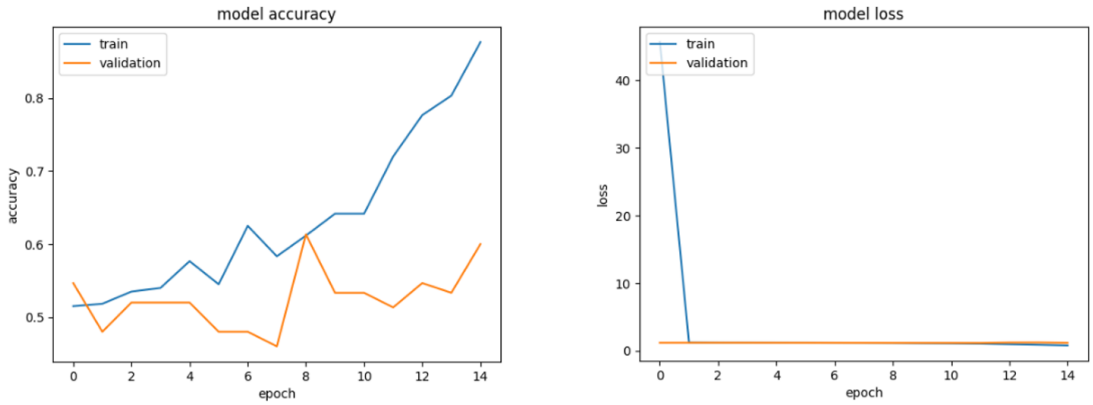

Figura 20. Evolución de precisión y pérdida a lo largo de épocas en el modelo original I

Los resultados del modelo con el conjunto de testing se muestran a continuación gracias a la función *Classification Report* y la matriz de confusión correspondiente [Figura 21]. Como aclaración, el valor 0 corresponderá a voces sintéticas mientras que 1 se referirá a voces reales.

	precision	recall	f1-score	support
0	0.55	0.74	0.63	125
1	0.61	0.40	0.48	125
accuracy			0.57	250
macro avg	0.58	0.57	0.56	250
weighted avg	0.58	0.57	0.56	250

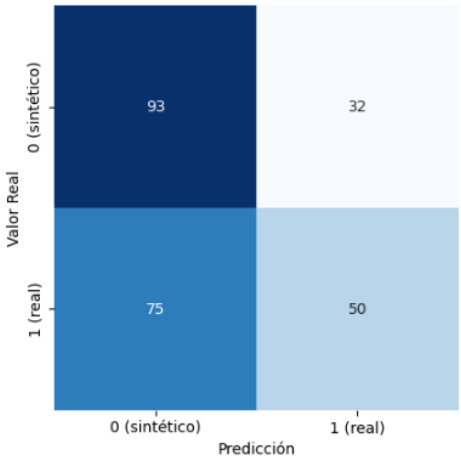

Figura 21. Matriz de confusión del modelo original I

4.2.2. Modelo original II (1 capa Conv2D)

Como segundo modelo se ha optado por desarrollar un modelo más simple que el primero debido a un posible sobreajuste que este ha podido sufrir según se puede ver en el análisis de sus resultados de validación. Este segundo contará con solamente una capa de convolución de dos dimensiones con tamaño de kernel de 3x3, una capa de MaxPooling de tamaño 2x2 para reducir la dimensionalidad de los datos, una capa de Dropout de tasa 0,4 y otra de 0,2 para reducir el sobreajuste y una capa plenamente conectada de 128 neuronas.

La función de activación permanecerá siendo softmax al igual la función de pérdida será binary_crossentropy y la función de optimización Adam. El modelo se entrenará durante 15 épocas. El código empleado para el modelo es el siguiente [Código 6].

```
# Modelo SIMPLE

model1 = Sequential()

model1.add(Input(shape=(128, 481,1)))
model1.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.4))
model1.add(Flatten())
model1.add(Dense(128, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Dense(2, activation='softmax'))
model1.summary()

model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model1.fit(train_features, train_labels_encoded, epochs=15, validation_spli
t=0.2)
```

Código 6. modelo original II

Los resultados para el entrenamiento del modelo han sido de 0,5 de precisión para el entrenamiento y 0,48 para la validación.

La evolución de las precisiones y la pérdida durante las distintas épocas se muestran a continuación [Figura 22].

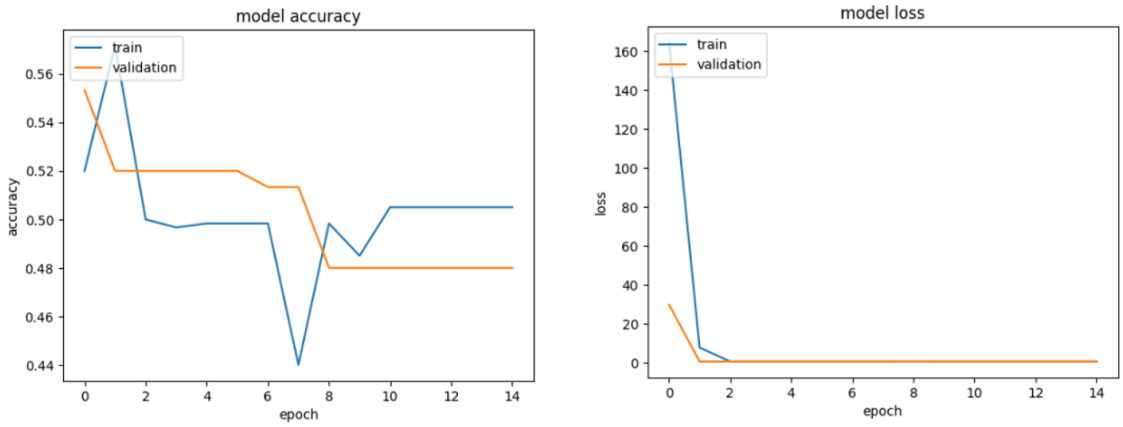

Figura 22. Evolución de precisión y pérdida a lo largo de épocas en el modelo original II

Los resultados del modelo con el conjunto de testing se muestran a continuación gracias a la función *Classification Report* y la matriz de confusión correspondiente [Figura 23].

	precision	recall	f1-score	support
0	0.00	0.00	0.00	125
1	0.50	1.00	0.67	125
accuracy			0.50	250
macro avg	0.25	0.50	0.33	250
weighted avg	0.25	0.50	0.33	250

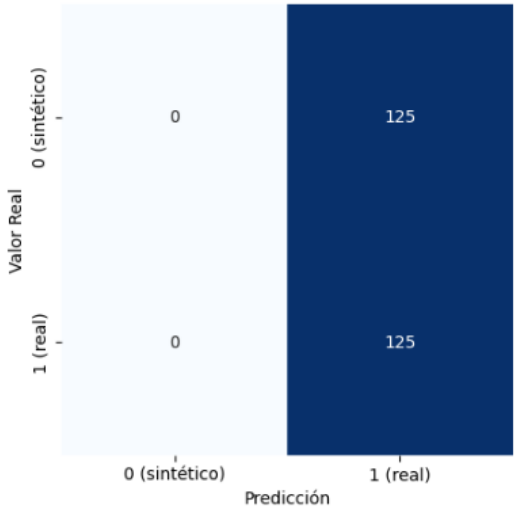

Figura 23. Matriz de confusión del modelo original II

4.2.3. Modelo VGG 16

Como primer modelo pre-entrenado se utilizará el conocido modelo VGG 16 propuesto originalmente por miembros de la Universidad de Oxford en 2014. Esta red neuronal contiene 16 capas de profundidad entre capas de convolución y plenamente conectadas [32] (véase Figura 24). Además, ha sido entrenada con más de un millón de imágenes ofrecidas por la base de datos de ImageNet [33]. Esta red ha sido utilizada para todo tipo de problemas de clasificación y se adaptará para poder utilizarse en el problema de clasificación binaria del trabajo. Debido a la complejidad del modelo se emplearán 5 épocas, manteniendo el resto de parámetros igual que los modelos anteriores.

Figura 24. Arquitectura VGG 16 [31]

Los resultados para el entrenamiento del modelo han sido de 0,97 de precisión para el entrenamiento y 0,76 para la validación. La evolución de las precisiones y la pérdida durante las distintas épocas se muestran en la Figura 25.

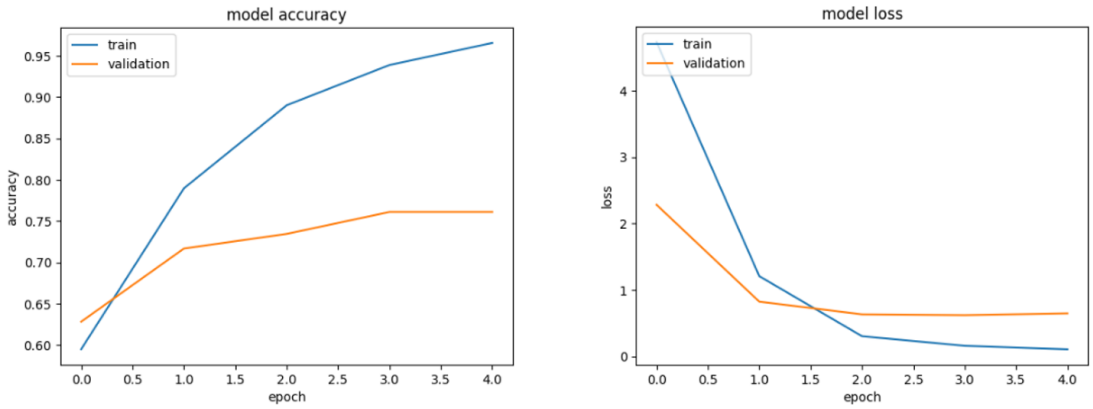

Figura 25. Evolución de precisión y pérdida a lo largo de épocas en el modelo VGG 16

Los resultados del modelo con el conjunto de testing se enseñan mediante el reporte de clasificación y la matriz de confusión [Figura 26].

	precision	recall	f1-score	support
0	0.71	0.58	0.63	125
1	0.64	0.76	0.70	125
accuracy			0.67	250
macro avg	0.67	0.67	0.67	250
weighted avg	0.67	0.67	0.67	250

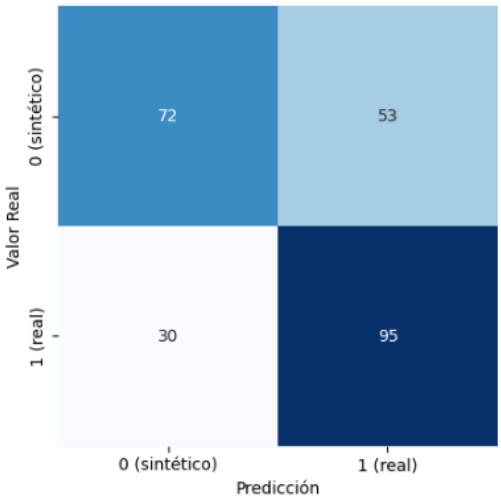

Figura 26. Matriz de confusión del modelo VGG 16

4.2.4. Modelo VGG 19

El modelo VGG 19 consiste en una versión más compleja del modelo VGG 16. Esta, como su nombre indica cuenta con 19 capas de profundidad y ha recibido el mismo entrenamiento que el modelo en el que se basa [34].

Los resultados para el entrenamiento del modelo han sido de 0,98 de precisión para el entrenamiento y 0,73 para la validación. La evolución de las precisiones y la pérdida durante las distintas épocas se muestran a continuación en la Figura 27.

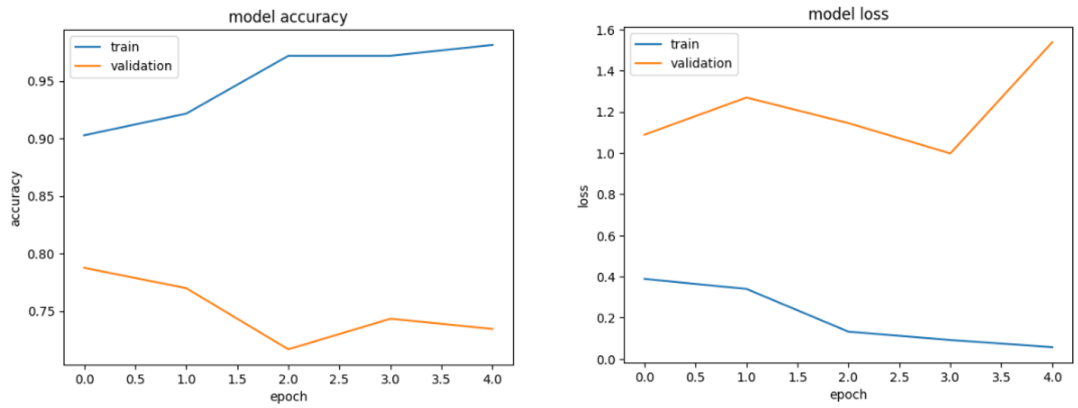

Figura 27. Evolución de precisión y pérdida a lo largo de épocas en el modelo VGG 19

Los resultados obtenidos del modelo al evaluar el conjunto de pruebas se presentan a continuación mediante el uso de la función "Classification Report" y la matriz de confusión [Figura 28].

	precision	recall	f1-score	support
0	0.72	0.46	0.57	125
1	0.61	0.82	0.70	125
accuracy			0.64	250
macro avg	0.67	0.64	0.63	250
weighted avg	0.67	0.64	0.63	250

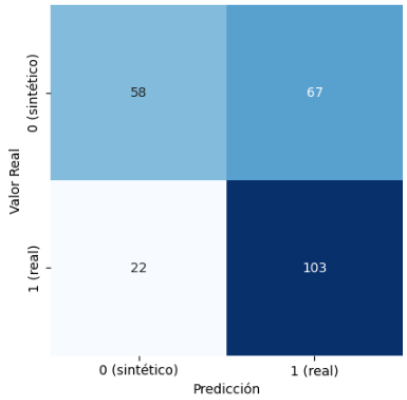

Figura 28. Matriz de confusión del modelo VGG 19

4.2.5. Modelo Inception

Inception consiste en un modelo para clasificación de imágenes entrenado al igual que los modelos VGG con el dataset de ImageNet. Ha demostrado conseguir un 78,1% de precisión en este conjunto de datos [35].

Los resultados para el entrenamiento del modelo han sido de 0,79 de precisión para el entrenamiento y 0,56 para la validación. La evolución de las precisiones y la pérdida durante las distintas épocas se muestran a continuación en la Figura 29.

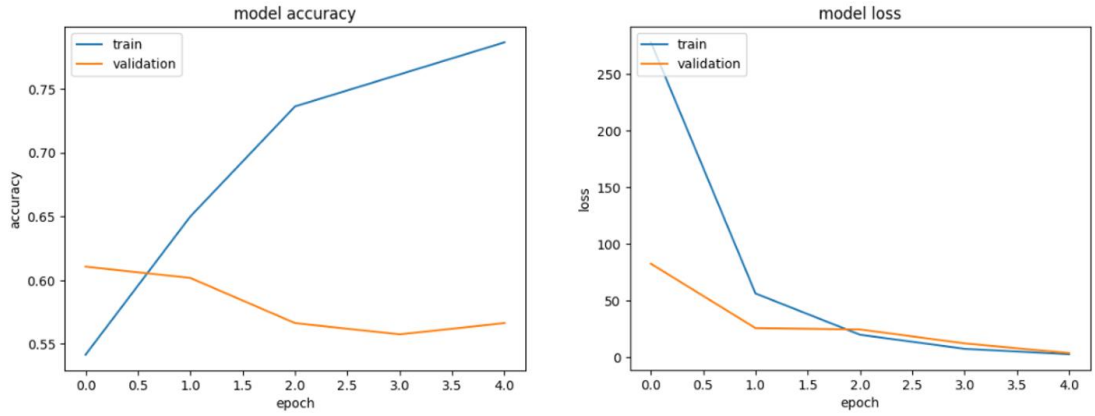

Figura 29. Evolución de precisión y pérdida a lo largo de épocas en el modelo Inception

Se presentan a continuación los resultados con el dataset de testing [Figura 30].

	precision	recall	f1-score	support
0	0.66	0.51	0.58	125
1	0.60	0.74	0.66	125
accuracy			0.62	250
macro avg	0.63	0.62	0.62	250
weighted avg	0.63	0.62	0.62	250

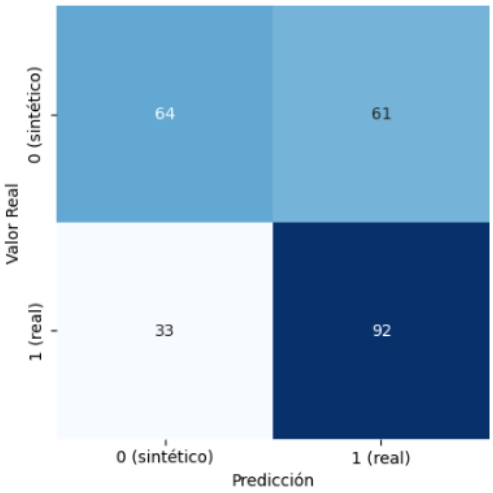

Figura 30. Matriz de confusión del modelo Inception

4.2.6. Modelo ResNet 50

ResNet50 consiste en un modelo preentrenado que cuenta con 50 capas de profundidad. También ha sido entrenado por la base de datos de ImageNet [36].

Los resultados para el entrenamiento del modelo han sido de 0,84 de precisión para el entrenamiento y 0,56 para la validación. La evolución de las precisiones y la pérdida durante las distintas épocas se muestran a continuación en la Figura 31.

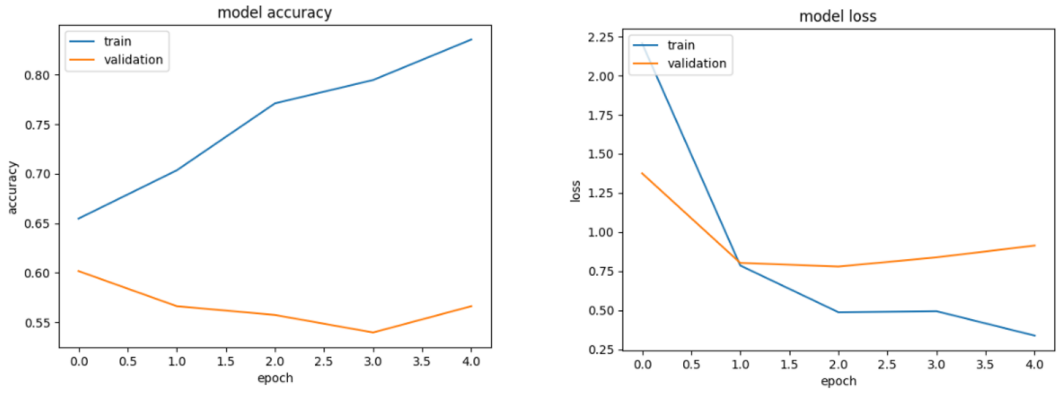

Figura 31. Evolución de precisión y pérdida a lo largo de épocas en el modelo ResNet50

Los resultados del modelo con el conjunto de testing se muestran a continuación [Figura 32].

	precision	recall	f1-score	support
0	0.68	0.63	0.66	125
1	0.66	0.70	0.68	125
accuracy			0.67	250
macro avg	0.67	0.67	0.67	250
weighted avg	0.67	0.67	0.67	250

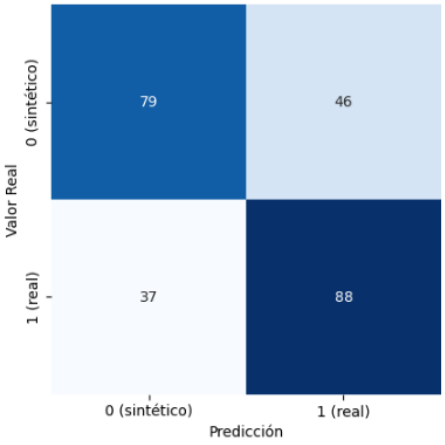

Figura 32. Matriz de confusión del modelo ResNet50

Capítulo 5. Análisis de resultados

5.1. Resultados de pruebas de seguridad con Smart Personal Assistants

En este apartado se han tomado los resultados extraídos de las pruebas de seguridad en asistentes personales realizadas previamente. Para ello este parte se dedica a realizar un análisis global de todos los resultados obtenidos, comprobar posibles diferencias entre los distintos asistentes, corroborar si las hipótesis iniciales planteadas son ciertas y finalizar extrayendo unas conclusiones acerca de la prueba en cuestión y qué se puede trazar a partir de estos resultados.

Al igual que en la sección de las pruebas, se ha dividido este apartado de análisis entre el estudio de las pruebas de suplantación y el relacionado con el posible acceso a aplicaciones comprometidas.

5.1.1. Ataques de suplantación de identidad por voz

Antes de realizar el análisis, se presentarán en la Tabla 7 un resumen de las pruebas realizadas en todos los asistentes y sus correspondientes resultados.

Tabla 7. Resumen de resultados de ataques de suplantación

Método de suplantación de voz	**Siri**	**Alexa**	**Google Assistant**
Grabación de voz	Sí es reconocida como la voz original	Sí es reconocida como voz de perfil creado	Sí es reconocida como persona conocida
Voz text-to-speech	No es reconocida	No es reconocida como voz de perfil	No es reconocida como persona conocida
Voz generada por inteligencia artificial	Sí es reconocida como la voz original	Sí es reconocida como voz de perfil creado	Sí es reconocida como persona conocida

Observando los resultados de las pruebas se puede comprobar que todos los asistentes poseen un comportamiento similar y por lo tanto se puede asegurar que sus sistemas de seguridad tienen un funcionamiento parecido. En el caso del ataque de suplantación con voces grabadas, todos los sistemas dan acceso al atacante, interpretándolo como la persona a identificar. Esta brecha de seguridad abre la puerta completamente a este tipo de ataques ya que, como se ha mencionado, únicamente es necesario contar con el comando de activación correspondiente de cada asistente para poder acceder a él, pudiendo cualquier otra voz decir las consultas tras pasar este control.

Por otra parte, los sistemas genéricos de voces Text-To-Speech no consiguen atravesar estos sistemas de seguridad como era de esperar, por lo que al menos puede asegurarse que los 3 dispositivos cuentan con un sistema de reconocimiento de voz que no comete errores graves y permite acceder al mismo a una voz completamente distinta a la del usuario original. Sin embargo, tras recrear voces con sistemas de inteligencia artificial (Resemble.ai en este caso) nuevamente todos los dispositivos tienen problemas para reconocerlo como una voz generada por ordenador.

En los tres asistentes, se ha podido acceder a los mismos pronunciando el comando de activación correspondiente mediante el sistema de clonación de voz. Este hecho es incluso más preocupante que el caso de la grabación de voces, ya estos sistemas de inteligencia artificial están viendo avances significativos y mediante una muestra cada vez más breve de la voz del usuario se producen modelos más precisos que permiten traspasar la seguridad de los asistentes sin ningún tipo de problema.

5.1.2. Acceso a aplicaciones comprometidas

A raíz de los claros problemas de seguridad que presentan los asistentes personales ante los ataques mencionados, se decidió realizar un estudio acerca de la dificultad que presentaban los distintos asistentes a la hora de acceder a determinadas aplicaciones comprometidas e información privada.

El resumen de los resultados se muestra en la Tabla 8.

Tabla 8. Resumen de resultados de pruebas de acceso a aplicaciones comprometidas

Tipo de aplicación	**Siri**	**Alexa**	**Google Assistant**
Banca y Finanzas	Requiere de desbloqueo de teléfono y credenciales de la aplicación	No hay opción a acceder a cuentas o realizar transacciones	No se dispone de este tipo de aplicaciones
Compras on-line	Esta funcionalidad no está permitida desde Siri	Posibilidad de comprar en Amazon mediante identificación de perfil	Funcionalidad restringida a Estados Unidos
Mensajería	Permite enviar mensajes y correos electrónicos desde la pantalla de bloqueo	Permite enviar mensajes a otros usuarios de Alexa	No se dispone de esta funcionalidad
Calendarios y tareas	Requiere de desbloqueo de teléfono	Permite acceder a la información de eventos y realizar modificaciones sin verificar el perfil	Permite acceder a la información de eventos y realizar modificaciones tras identificación previa

Los resultados presentan notas positivas y negativas en cada asistente. La información que cuenta la mayor seguridad posible es la relacionada con las cuentas bancarias y el hecho de realizar posibles transacciones ya que o bien directamente no presentan la opción de acceder a este tipo de datos (Alexa, Google Assistante) o requieren de varias identificaciones adicionales (Siri).

En lo respectivo a compras en línea el único asistente que cuenta con esta funcionalidad es Alexa, mediante la compra en Amazon. Esta opción acarrea graves problemas de seguridad ya que no requiere de capas de seguridad adicionales de manera predeterminada, por lo que mediante suplantaciones de voces como las mostradas se pueden adquirir productos usando este asistente.

Si bien Siri es el asistente que más tiende a solicitar verificaciones adicionales para el uso de sus aplicaciones, cuando se trata de mensajería no presenta ninguna seguridad extra para enviar mensajes ya sean mensajes de texto, mensajes de WhatsApp o correos electrónicos. Por lo tanto, con acceder al uso del asistente, un atacante puede enviar mensajes no deseados y puede comunicarse con contactos privados del usuario. Alexa también permite enviar mensajes, pero

únicamente a otros usuarios de Alexa, por lo que las opciones de comunicación se reducen, pero también se abre una brecha de seguridad en dicho asistente.

Por último, de cara a acceso de eventos y datos del calendario del usuario, tanto Alexa como Google Assitant permiten que el atacante, en caso de traspasar la seguridad inicial, pueda acceder a información privada acerca de la agenda del usuario, así como establecer falsos eventos si se quiere.

5.2. Resultados de modelos de clasificación de voces reales y voces sintéticas

En esta sección se pondrán en conjunto los resultados de los modelos tanto en la parte de entrenamiento como en la parte de testing. Se valorarán dichos resultados, estudiando cuál ha sido el comportamiento de los modelos y tratando de vislumbrar cuáles han sido las características sobre las que se han basado las redes neuronales a la hora de realizar la clasificación.

Tras comprobar el rendimiento obtenido se compararán los resultados con los recogidos en una encuesta realizada a personas reales, y se valorarán en perspectiva los modelos desarrollados.

5.2.1. Valoración de modelos

A continuación, se presentan los resultados obtenidos en los 6 modelos empleados en la Tabla 9.

Tabla 9. Precisión de los distintos modelos de clasificación

	Train	Test
Modelo Original I (2 Conv2D)	76%	57%
Modelo Original II (1 Conv2D)	50%	50%
VGG 16	97%	67%
VGG 19	98%	64%
Inception	79%	62%
ResNet 50	84%	67%

A partir de los resultados de precisión de clasificación mostrados se pueden extraer varios apuntes. En primer lugar, se aprecia una clara mejora en valores de precisión por parte de los modelos pre-entrenados en comparación con los modelos realizados desde cero. Esto seguramente se debe a la complejidad de estos modelos, ya que se componen de grandes números de capas. Esta hipótesis se ve reflejada en los resultados del modelo original II, que siendo el más simple de todos ofrece valores muy bajos de precisión, siendo prácticamente una predicción aleatoria la que realiza, lo

que se puede interpretar como que contar con una sola capa es insuficiente para abordar este tipo de problemas.

En los valores de precisión durante el entrenamiento se aprecian en general buenos valores de precisión lo que indica que el modelo ha sido capaz de extraer características de los datos utilizados como entrada para realizar después la clasificación. A pesar de ello, los valores de precisión de la parte de testing son bajos, tratándose este de un problema de clasificación binaria. Se interpreta que los modelos han podido sufrir de cierto sobreajuste lo que ha hecho que las características sobre las que se ha basado el desarrollo de los modelos estuviesen presentes en los datos de entrada, pero no tuviesen tanta presencia en los datos de entrenamiento. Este sesgo se ha podido deber al tamaño de los datos empleados, ya que es posible que de haber trabajado con mayores cantidades de datos, las características extraídas pudiesen haber sido distintas.

Adicionalmente, para valorar estos resultados de precisión se realizó una encuesta con personas reales para comprobar cuánto difieren las predicciones humanas de las realizadas por los modelos. Para realizar esta consulta rápida se pidió a diez personas distintas que escuchasen 12 audios distintos extraídos del dataset utilizado en los modelos y tratasen de clasificarlos según si eran voces reales y voces generadas por ordenador. Los cuatro primeros audios fueron erróneamente clasificados por el mejor modelo (VGG 16) mientras que los ocho siguientes han sido escogidos aleatoriamente. Los resultados se muestran en la Figura 33.

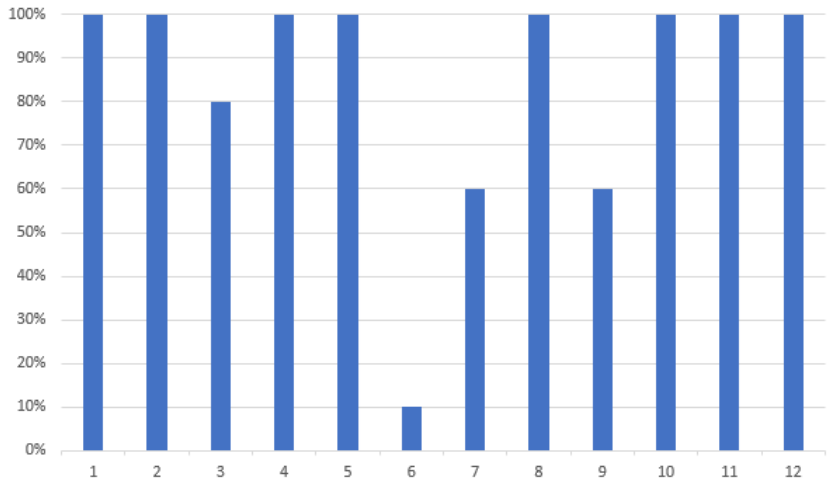

Figura 33. Resultados de precisión en clasificación de voces reales o sintéticas realizada por personas

La precisión total de la encuesta fue de 86,66%.

Como se puede ver, el ser humano también puede tener problemas a la hora de clasificar este tipo de audios por lo que los resultados de los modelos, a pesar de que siguen siendo por debajo de lo esperado, no son tan deficientes si tenemos en cuenta que se trata de un problema en el que las personas también tienen cierta cuota de error.

Curiosamente, las predicciones por parte de las personas en los audios que había clasificado erróneamente el mejor modelo desarrollado (4 primeros audios) tuvieron muy buenos resultados de acierto, siendo el resto de los audios los que aportaron mayores errores.

Esto se puede explicar debido a que la extracción de características por parte de los modelos empleados difiere de los criterios que emplean las personas para diferenciar audios reales de audios generados por ordenador. Al emplear los espectrogramas como entrada en el caso de los modelos y las personas únicamente escuchar el audio sin contar con la representación visual, es de esperar que los parámetros utilizados por cada uno sean distintos. Se explorará esta cuestión más en profundidad en el siguiente apartado y se buscará reconocer cuál ha sido el criterio empleado por parte de los modelos.

5.2.2. Análisis de extracción de características

Para abordar el análisis de las características extraídas por los modelos, se ha decidido contar como referencia con el modelo que mejores valores de precisión ha ofrecido, el cual es el modelo VGG 16. A continuación, se ofrecerán espectrogramas de audios que han sido catalogados como voces sintéticas o reales, mostrando tanto predicciones correctas como incorrectas [Figura 34] [Figura 35]. A partir de ello se hará un análisis visual de las imágenes para tratar de explicar el criterio de selección elegido por el modelo y posibles motivos por los que los valores de precisión no han sido los esperados

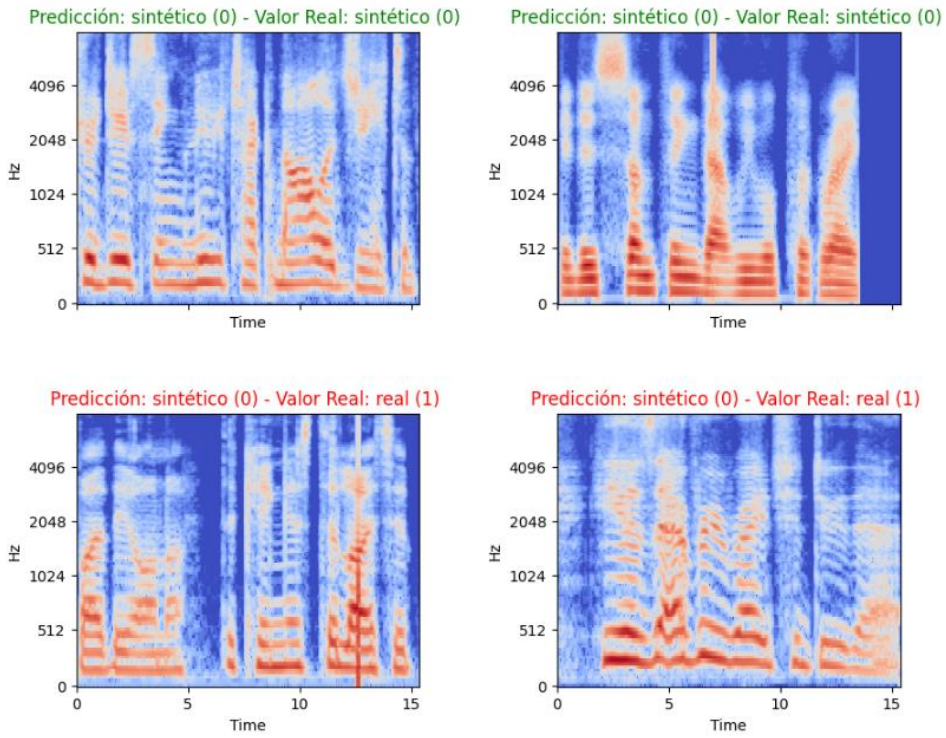

Figura 34. Espectrogramas de Mel de audios catalogados como voz sintética

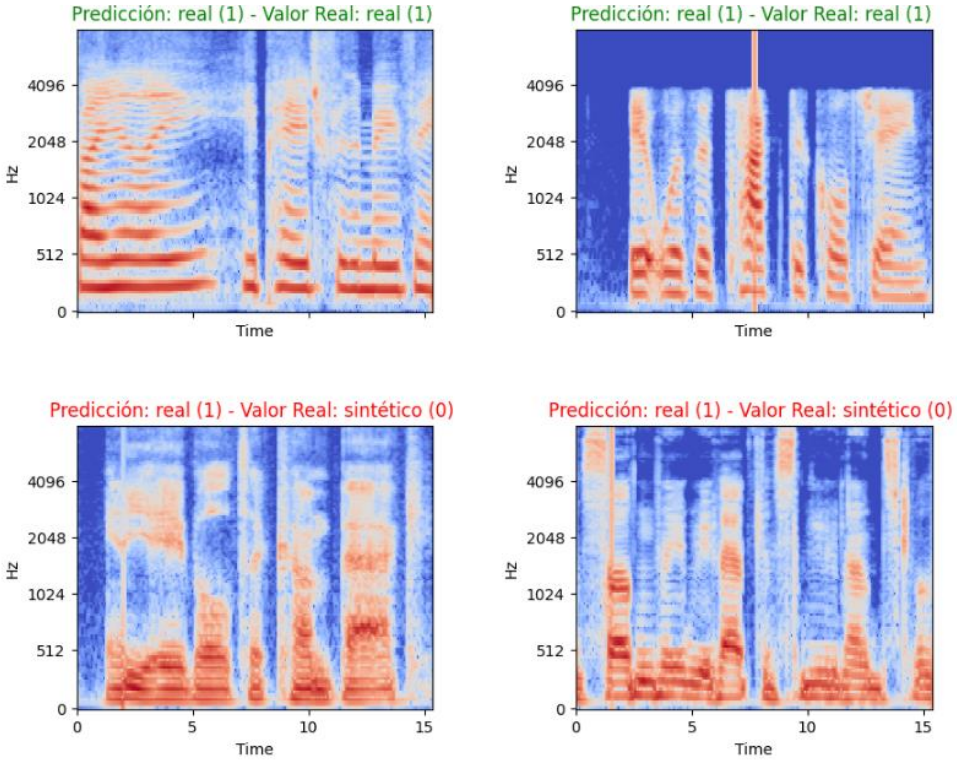

Figura 35. Espectrogramas de Mel de audios catalogados como voz real

Si se observan los espectrogramas en detalle se pueden intuir una serie de características por las cuales el modelo ha decidido guiar su clasificación. En el caso de aquellas voces reconocidas como sintéticas se pueden apreciar un mayor número de interrupciones en el eje temporal que aquellas reconocidas como voces reales. Estas pausas parecen ser más abruptas en el caso de las voces sintéticas y cuando aparecen en las voces humanas, suelen ser de mayor duración. Esto puede deberse a que las voces sintéticas tienden a tener pausas menos naturales y se producen más a menudo entre palabras.

Se puede ver también que, en el caso de las voces sintéticas, la mayoría de estas cuentan con mayor presencia en frecuencias altas. A pesar de ello, el modelo no se ha debido guiar demasiado por este criterio ya que se aprecian algunas predicciones erróneas en las que hubiesen podido ser distintas de seguir este sesgo.

Además, parece que las voces reales han sido identificadas en parte por tener una pausa inicial en algunas de ellas, lo cual no se encuentra en la mayoría de aquellas catalogadas como sintéticas. Esto claramente es una característica errónea sobre la cual se ha podido desarrollar el modelo y la cual ha podido afectar a la precisión del mismo.

Como se ha visto, parece que las características por las que se ha podido guiar el modelo son en parte correctas e incorrectas a la vez. Al ser algunos audios muy evidentes que son generados por ordenador y otros ser voces clonadas muy logradas, el modelo ha podido tener problemas (al igual que las personas) a la hora de determinar características con las que diferenciarlas de las voces reales. El objetivo del desarrollo de los modelos nunca ha sido conseguir altos niveles de precisión sino mostrar que las redes neuronales convolucionales y el uso de espectrogramas de Mel son un buen acercamiento inicial para abordar este tipo de problemas de clasificación, lo cual ha quedado comprobado viendo los resultados del modelo. A pesar de ello, existe un margen de mejora evidente en los modelos, que apuntan a requerir de una mayor complejidad y quizás necesiten ayudarse de otro tipo de redes para lograr una mejor extracción de características de las imágenes, punto que se ha apreciado claramente en esta última sección.

Capítulo 6. Conclusiones y trabajos futuros

6.1. Conclusiones de las pruebas de seguridad con Smart Personal Assistants

Las diferentes pruebas realizadas han permitido evidenciar los problemas de seguridad presentes en los Smart Personal Assistants, lo cual ya se había adelantado previamente. Los resultados ofrecidos por la sección de ataques de suplantación de identidad son especialmente preocupantes ya que todos los asistentes testeados están expuestos a este tipo de ataques ya sea mediante grabaciones o voces generadas por inteligencia artificial.

Se ha demostrado la facilidad con la que un atacante puede acceder a este tipo de sistemas sin herramientas demasiado complejas, lo que hace ver que es indispensable una revisión de los sistemas de autenticación y reconocimiento de voz de los SPA. La implementación de clasificadores potentes de voces reales y sintéticas es vital para un futuro seguro de estos asistentes ya que las herramientas de clonación de voz y los ataques relacionados serán cada vez más indetectables a medida que pase el tiempo.

El análisis de acceso a aplicaciones comprometidas en los asistentes personales tampoco proporciona un mensaje completamente esperanzador. A pesar de que muchos dispositivos cuentan con varias verificaciones o directamente no permiten acceder a información comprometida, lo cierto es que cada dispositivo analizado permitía un uso y acceso sencillo a algunas de estas aplicaciones estudiadas (Siri - mensajería, Alexa - compras online, Asistente de Google – Calendarios y eventos). Siguiendo la línea de lo mencionado respecto a las brechas de seguridad de los asistentes, se puede confirmar que acceder a información privada o comprometida es relativamente sencillo en estos sistemas. Además, estos accesos han sido los más básicos que permitían los dispositivos, pero un atacante especializado que pueda traspasar capas de seguridad o pueda acceder de manera más sencilla a determinadas aplicaciones puede suponer un peligro real para la seguridad y privacidad del usuario del asistente personal.

Es por esto que es indispensable reforzar la seguridad en todas estas aplicaciones de información comprometida, ya sea mediante códigos, contraseñas o sistemas de doble verificación. Como

individuo, se ha de tener en cuenta la existencia de estas amenazas y se han de establecer los mayores parámetros de seguridad que permite el dispositivo que se utilice.

6.2. Conclusiones de los modelos de clasificación de voces reales y sintéticas

El desarrollo de modelos de redes neuronales convolucionales para clasificar voces reales y voces sintéticas ha dejado varias lecturas a destacar. A pesar de que los niveles de precisión de los modelos no han superado el 70% utilizando los datos de entrenamiento, se ha comprobado que este tipo de redes y el uso de espectrogramas de Mel como objeto de entrada son unas buenas aproximaciones a la hora de atacar este problema. Además, se ha demostrado que esta clasificación en concreto es un problema que incluso puede resultar complicado al ser humano, realizando los individuos encuestados varias predicciones erróneas. Esto permite rebajar el listón de los baremos de precisión esperados por parte de los modelos, ya que realmente existen voces sintéticas que engañan al propio humano.

Por otro lado, también se ha demostrado que las características extraídas por los modelos para su desarrollo pueden no estar directamente relacionadas con la categoría a la que pertenezca la voz, sino con otros elementos como las pausas y la duración de las mismas. Esto debe ser un matiz a anotar de cara a plantear futuros modelos y tratar de conseguir mejores valores de precisión en ellos.

6.3. Trabajos futuros

Durante el desarrollo de este trabajo, se ha podido reflexionar acerca de futuras vías de investigación derivadas de los campos planteados. A continuación, se proponen líneas de investigación a partir tanto de Smart Personal Assistants como del desarrollo de modelos de clasificación de voces reales y sintéticas.

6.3.1. Propuestas en Smart Personal Assistants

- Detección de comandos fraudulentos: un apartado interesante a investigar acerca de la seguridad de los asistentes personales son los ataques mediante comandos parecidos a los originales pero que permiten al atacante acceder al dispositivo. Este tipo de ataques como es el Voice Squatting mencionado en la sección de estado del arte, pueden suponer la activación de software malicioso sin que el usuario se dé cuenta. Es por ello que sería relevante investigar acerca de posibles sistemas de detección de este tipo de ataques para tratar de mitigarlos y reforzar la seguridad de los asistentes personales.

- Seguridad en entornos IoT: hacer una profunda investigación de la seguridad de los SPAs en el contexto de Internet de las cosas puede contribuir enormemente a reforzar la seguridad en dispositivos que hoy en día forman parte de la casa, y que se encuentran conectados todos a raíz del mismo asistente. Investigar la interacción entre estos y estudiar posibles brechas de seguridad existentes aportaría una perspectiva nueva a la seguridad de los Smart Personal Assistants.

6.3.2. Propuestas en clasificación de voces reales y sintéticas

- Uso o combinación de nuevos tipos de modelos: si bien las redes neuronales convolucionales son herramientas que se ha demostrado que ofrecen un buen funcionamiento en problemas de clasificación, una opción para mejorar la precisión de los modelos es la combinación de este tipo de redes con otras similares como Encoders [30] o utilizar estas últimas de manera separada.

- Prueba con nuevos datasets y preprocesado de audios: de cara a complementar el trabajo realizado en este proyecto pueden incorporarse nuevos datos de audios como voces generadas por inteligencia artificial más avanzadas. Además, es interesante investigar nuevas técnicas para el preprocesado de audio que puedan modificar las voces de tal manera que se facilite más aún el entrenamiento del modelo en cuestión.

Bibliografía

[1] C. V. Amores, «Análisis y Evaluación de Sistemas de Autenticación para Smart Personal Assistants,» 2021. [En línea]. Disponible en: https://www.researchgate.net/publication/352277878_Analisis_de_Seguridad_y_Privacidad_en_dispositivos_de_la_Internet_de_las_Cosas_usados_por_jovenes. [Último acceso: 2 Julio 2023].

[2] K. O'Flaherty, «Amazon Staff Are Listening To Alexa Conversations -- Here's What To Do,» 2019. [En línea]. Disponible en: https://www.forbes.com/sites/kateoflahertyuk/2019/04/12/amazon-staff-are-listening-to-alexa-conversations-heres-what-to-do/#69b0f95771a2. [Último acceso: 29 Junio 2023].

[3] IDR&D, «What is Voice Biometrics and why should you use it?,» [En línea]. Disponible en: https://www.idrnd.ai/voice-biometrics/. [Último acceso: 25 Junio 2023].

[4] Plumvoice, «What Is Voice Biometrics?,» [En línea]. Disponible en: https://www.plumvoice.com/resources/blog/voice-biometrics/#:~:text=Voice%20biometrics%20technology%20verifies%20the,%E2%80%9Cvoice%20biometrics%E2%80%9D%20in%20general.. [Último acceso: 25 Junio 2023].

[5] J. Umawing, «Researchers discover vulnerabilities in smart assistants' voice commands,» *Malware Bytes,* 2018.

[6] K. N. Yip, «Top 10 Threats from Unprotected Intelligent Personal Assistants (IPAs),» 2017. [En línea]. Disponible en: https://resources.infosecinstitute.com/topic/top-10-threats-unprotected-intelligent-personal-assistants-ipas/. [Último acceso: 27 Junio 2023].

[7] IDRND, «What is Voice Cloning?,» [En línea]. Disponible en: https://www.idrnd.ai/what-is-voice-cloning/. [Último acceso: 26 Junio 2023].

[8] Microsoft, «VALL-E,» 2023. [En línea]. Disponible en: https://vall-e.io/. [Último acceso: 20 Junio 2023].

[9] A. Bunn, «Artificial Imposters—Cybercriminals Turn to AI Voice Cloning for a New Breed of Scam,» 2023. [En línea]. Disponible en: https://www.mcafee.com/blogs/privacy-identity-protection/artificial-imposters-cybercriminals-turn-to-ai-voice-cloning-for-a-new-breed-of-scam/. [Último acceso: 20 Junio 2023].

[10] Dessa News, «Detecting Audio Deepfakes With AI,» 2019. [En línea]. Disponible en: https://medium.com/dessa-news/detecting-audio-deepfakes-f2edfd8e2b35. [Último acceso: 22 Junio 2023].

[11] Dessa News, «RealTalk (Pt. II): How We Recreated Joe Rogan's Voice Using AI,» 2019. [En línea]. Disponible en: https://medium.com/dessa-news/realtalk-how-it-works-94c1afda62f0. [Último acceso: 15 Junio 2023].

[12] E. Deruty, «Intuitive understanding of MFCCs,» 2022. [En línea]. Disponible en: https://medium.com/@derutycsl/intuitive-understanding-of-mfccs-836d36a1f779. [Último acceso: 2 Julio 2023].

[13] Wikipedia, «Spectral Centroid,» [En línea]. Disponible en: https://en.wikipedia.org/wiki/Spectral_centroid. [Último acceso: 18 Junio 2023].

[14] C. Lewis, «How to Create & Understand Mel-Spectrograms,» 2021. [En línea]. Disponible en: https://importchris.medium.com/how-to-create-understand-mel-spectrograms-ff7634991056. [Último acceso: 2 Julio 2023].

[15] S. Duda, «Urban Environmental Audio Classification Using Mel Spectrograms,» 2020. [En línea]. Disponible en: https://scottmduda.medium.com/urban-environmental-audio-classification-using-mel-spectrograms-706ee6f8dcc1. [Último acceso: 2 Julio 2023].

[16] M. Mishra, «Convolutional Neural Networks, Explained,» 2020. [En línea]. Disponible en: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939. [Último acceso: 1 Julio 2023].

[17] S. Saha, «A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,» 2018. [En línea]. Disponible en: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/. [Último acceso: 1 Julio 2023].

[18] X. W. Junichi Yamagishi, «ASVspoof 2021: accelerating progress in spoofed and deepfake speech detection,» 2021. [En línea]. Disponible en: https://www.isca-speech.org/archive/pdfs/asvspoof_2021/yamagishi21_asvspoof.pdf. [Último acceso: 27 Junio 2023].

[19] Resemble.ai, «Resemble.ai,» [En línea]. Disponible en: https://www.resemble.ai/. [Último acceso: 20 Junio 2023].

[20] «Librosa,» [En línea]. Disponible en: https://librosa.org/doc/latest/index.html. [Último acceso: 15 Junio 2023].

[21] «Tensorflow,» [En línea]. Disponible en: Tensorflow.org. [Último acceso: 15 Junio 2023].

[22] «skicit - learn,» [En línea]. Disponible en: https://scikit-learn.org/stable/. [Último acceso: 15 Junio 2023].

[23] «Cloud Text-to-Speech,» [En línea]. Disponible en: https://cloud.google.com/text-to-speech?hl=es. [Último acceso: 15 Junio 2023].

[24] Britta O'Boyle, «What is Siri and how does Siri work?,» *Pocket-link,* 2021.

[25] «Apple,» [En línea]. Disponible en: https://support.apple.com/es-es/HT204389. [Último acceso: 17 Junio 2023].

[26] M. Terán, «Comprar con Siri: por qué Apple ha decidido no implementarlo aún,» 2022. [En línea]. Disponible en: https://www.eleconomista.es/tecnologia/noticias/11721794/04/22/Comprar-con-Siri-porque-Apple-ha-decidido-no-implementarlo-aun-.html. [Último acceso: 28 Junio 2023].

[27] Y. Fernández, «Qué es Alexa, qué puedes hacer con él y qué dispositivos son compatibles,» 2023. [En línea]. Disponible en: https://www.xataka.com/basics/que-alexa-que-puedes-hacer-que-dispositivos-compatibles. [Último acceso: 28 Junio 2023].

[28] «Amazon. Dispositivos Echo y Alexa,» [En línea]. Disponible en: https://www.amazon.es/b?ie=UTF8&node=15752872031.

[29] M. Martínez, «Guía de Google Assistant: configuración, usos y trucos,» 2022. [En línea]. Disponible en: https://www.movilzona.es/tutoriales/software/google-assistant/. [Último acceso: 25 Junio 2023].

[30] «Kaggle. Fake Speech Detection: Conformer [TF],» [En línea]. Disponible en: https://www.kaggle.com/code/awsaf49/fake-speech-detection-conformer-tf#9.-Data-Pipeline-%F0%9F%8D%9A. [Último acceso: 1 Julio 2023].

[31] A. Rubiales, «Funciones de error con Entropia: Cross Entropy y Binary Cross Entropy,» 2021.

[32] «VGG: ¿Qué es este modelo? ¡Daniel te lo cuenta todo!,» 2023. [En línea]. Disponible en: https://datascientest.com/es/vgg-que-es-este-modelo-daniel-te-lo-cuenta-todo. [Último acceso: 3 Julio 2023].

[33] «Mathworks vgg16,» [En línea]. Disponible en: https://es.mathworks.com/help/deeplearning/ref/vgg16.html#bvo3twr-1_sep_mw_6dc28e13-2f10-44a4-9632-9b8d43b376fe. [Último acceso: 1 Julio 2023].

[34] «Mathworks vgg19,» [En línea]. Disponible en: https://es.mathworks.com/help/deeplearning/ref/vgg19.html. [Último acceso: 1 Julio 2023].

[35] «Google Cloud Guía avanzada de Inception v3,» [En línea]. Disponible en: https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=es-419. [Último acceso: 1 Julio 2023].

[36] «MathWorks resnet,» [En línea]. Disponible en: https://es.mathworks.com/help/deeplearning/ref/resnet50.html. [Último acceso: 1 Julio 2023].

[37] P. M. R. Española, «17 Objetivos de Desarrollo Sostenible (ODS) para transformar el mundo,» [En línea]. Disponible en: https://www.pactomundial.org/que-puedes-hacer-tu/ods/. [Último acceso: 3 Julio 2023].

Anexo I: Objetivos de Desarrollo Sostenible (ODS)

Los Objetivos de Desarrollo Sostenible (ODS) o Agenda 2030 son una iniciativa global creada por las Naciones Unidas en septiembre de 2015. Estos objetivos consisten en un conjunto de 17 metas interconectadas y desafiantes que buscan abordar los problemas más apremiantes que enfrenta tanto nuestro planeta como la humanidad en su conjunto [37] (véase **¡Error! No se encuentra el origen de la referencia.**).

Figura 36. Objetivos de Desarrollo Sostenible

Respecto a los objetivos de desarrollo sostenible, el desarrollo de este TFG se alinea con varios de estos. Para empezar, apoya el objetivo 4 de tener una Educación de Calidad, al promover nuevas tecnologías que, con un correcto desarrollo, pueden ser una pieza fundamental para la educación del futuro. También sigue la línea del objetivo 9 de Industria, Innovación e Infraestructura, al consistir en un proyecto de evolución y mejora de una innovación tecnológica como son los asistentes personales y sus sistemas de clasificación de voces.

Además, el trabajo funciona acorde al objetivo 11 de Ciudades y Comunidades Sostenibles ya que se promueve el funcionamiento de dispositivos que requieren poca energía y no tienen problemas de contaminación. Estos dispositivos de reconocimiento de voz siguen la línea de la aparición de las ciudades inteligentes y formarán una parte clave de estas en el futuro, manteniendo un desarrollo sostenible y moderno de estas.

Anexo II: Código del desarrollo de modelos de clasificación

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
import IPython.display as ipd
from IPython.display import Audio


import tensorflow as tf, re, math
import tensorflow.keras.backend as K
import tensorflow_io as tfio
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import models, layers, regularizers
from tensorflow.keras.models import Sequential
from keras.models import Model, load_model, Sequential
from keras.layers import Dense, GlobalAveragePooling2D, Input, Flatten, Conv2D, Batch
Normalization, MaxPooling2D, Dropout, Concatenate
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.efficientnet import EfficientNetB7
from tensorflow.keras.applications.efficientnet import EfficientNetB0

from sklearn.preprocessing import normalize
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, roc_curve, accuracy_score, classification_
report, confusion_matrix
from sklearn.metrics import confusion_matrix as sk_confusion_matrix
from sklearn.preprocessing import LabelBinarizer
from sklearn.utils import shuffle

from PIL import Image
import plotly.graph_objects as go

## PARAMETROS AUDIO

sample_rate = 16000
duration = 1.5 #segundos
audio_len = int(sample_rate * duration)


# PARÁMETROS ESPECTROGRAMA
spec_freq = 128
n_fft = 512
hop_length = 50

# Lectura de datos

BASE_PATH = './Data'

train_df = pd.read_csv(f'{BASE_PATH}/keys/LA//CM/trial_metadata.txt',
                       sep=" ", header=None)
train_df.columns
=['speaker_id','trial_id','codec','trans','spoof_attack','class','trim','subset']
train_df = train_df.drop(columns=['codec','trans','spoof_attack','trim','subset'])
train_df['filepath'] =
```

```
f'{BASE_PATH}/ASVspoof2021_LA_eval/flac/'+train_df.trial_id+'.flac'
train_df['filepath'] = train_df['filepath'].astype(str)


# Data Augmentation

# Tools
def random_int(shape=[], minval=0, maxval=1):
    return tf.random.uniform(shape=shape, minval=minval, maxval=maxval, dtype=tf.int3
2)


def random_float(shape=[], minval=0.0, maxval=1.0):
    rnd = tf.random.uniform(shape=shape, minval=minval, maxval=maxval, dtype=tf.float
32)
    return rnd

# Trim
def TrimAudio(audio, epsilon=0.15):
    pos  = tfio.audio.trim(audio, axis=0, epsilon=epsilon)
    audio = audio[pos[0]:pos[1]]
    return audio

# Crop or Pad audio to keep a fixed length
def CropOrPad(audio, target_len, pad_mode='constant'):
    audio_len = tf.shape(audio)[0]
    if audio_len < target_len: # if audio_len is smaller than target_len then use Pad
ding
        diff_len = (target_len - audio_len)
        pad1 = random_int([], minval=0, maxval=diff_len) # select random location for
padding
        pad2 = diff_len - pad1
        pad_len = [pad1, pad2]
        audio = tf.pad(audio, paddings=[pad_len], mode=pad_mode) # apply padding
    elif audio_len > target_len:  # if audio_len is larger than target_len then use C
ropping
        diff_len = (audio_len - target_len)
        idx = tf.random.uniform([], 0, diff_len, dtype=tf.int32) # select random loca
tion for cropping
        audio = audio[idx: (idx + target_len)]
    audio = tf.reshape(audio, [target_len])
    return audio

# Normalize
def Normalize(data):
    MEAN = tf.math.reduce_mean(data)
    STD = tf.math.reduce_std(data)
    data = tf.math.divide_no_nan(data - MEAN, STD)
    return data


# Apply random noise to audio data
def GaussianNoise(audio, std=[0.0025, 0.025], prob=0.5):
    std = random_float([], std[0], std[1])
    if random_float() < prob:
        GN = tf.keras.layers.GaussianNoise(stddev=std)
        audio = GN(audio, training=True) # training=False don't apply noise to data
    return audio


# Randomly shift audio -> any sound at <t> time may get shifted to <t+shift> time
def TimeShift(audio, prob=0.5):
    if random_float() < prob:
        shift = random_int(shape=[], minval=0, maxval=tf.shape(audio)[0])
        if random_float() < 0.5:
```

```
            shift = -shift
        audio = tf.roll(audio, shift, axis=0)
    return audio

# Randomly mask data in time and freq axis
def TimeFreqMask(spec, time_mask, freq_mask, prob=0.5):
    if random_float() < prob:
        spec = tfio.audio.freq_mask(spec, param=freq_mask)
        spec = tfio.audio.time_mask(spec, param=time_mask)
    return spec


# Reparto de muestras

grupo_spoof = train_df[train_df['class'] == 'spoof']
grupo_bonafide = train_df[train_df['class'] == 'bonafide']

muestra_spoof = grupo_spoof.sample(500, replace=True, random_state=42)
muestra_bonafide = grupo_bonafide.sample(500, replace=True, random_state=42)

muestra = pd.concat([muestra_spoof,muestra_bonafide])
muestra = muestra.sample(frac=1, random_state=42)


# Prueba
# AUDIO FALSO
path_t = "./Data/ASVspoof2021_LA_eval/flac/LA_E_6012786.flac"

y, sr = librosa.load(path_t, sr=sample_rate)

ipd.Audio(y, rate=sr)


plt.figure(figsize=(20, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveplot', fontdict=dict(size=18))
plt.xlabel('Time', fontdict=dict(size=15))
plt.ylabel('Amplitude', fontdict=dict(size=15))
plt.show()



# Preprocesado audio

y = TrimAudio(y)
y = CropOrPad(y, audio_len)
y = TimeShift(y)
y = y.numpy()
#y = GaussianNoise(y)
#y = y.numpy()

plt.figure(figsize=(20, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveplot', fontdict=dict(size=18))

plt.xlabel('Time', fontdict=dict(size=15))
plt.ylabel('Amplitude', fontdict=dict(size=15))
plt.show()

# Creating a Discrete-Fourier Transform with our FFT algorithm
fast_fourier_transf = np.fft.fft(y)
# Magnitudes indicate the contribution of each frequency
magnitude = np.abs(fast_fourier_transf)
# mapping the magnitude to the relative frequency bins
frequency = np.linspace(0, sr, len(magnitude))
# We only need the first half of the magnitude and frequency
```

```
left_mag = magnitude[:int(len(magnitude)/2)]
left_freq = frequency[:int(len(frequency)/2)]
plt.plot(left_freq, left_mag)
plt.title('Discrete-Fourier Transform', fontdict=dict(size=15))
plt.xlabel('Frequency', fontdict=dict(size=12))
plt.ylabel('Magnitude', fontdict=dict(size=12))
plt.show()



# Short-time Fourier Transformation on our audio data
audio_stft = librosa.core.stft(y, hop_length=hop_length, n_fft=n_fft)
# gathering the absolute values for all values in our audio_stft
spectrogram = np.abs(audio_stft)
# Plotting the short-time Fourier Transformation
plt.figure(figsize=(20, 5))
# Using librosa.display.specshow() to create our spectrogram
librosa.display.specshow(spectrogram, sr=sr, x_axis='time',        y_axis='hz', hop_l
ength=hop_length)
plt.colorbar(label='Amplitude')
plt.title('Spectrogram (amp)', fontdict=dict(size=18))
plt.xlabel('Time', fontdict=dict(size=15))
plt.ylabel('Frequency', fontdict=dict(size=15))
plt.show()



# Short-time Fourier Transformation on our audio data
audio_stft = librosa.core.stft(y, hop_length=hop_length, n_fft=n_fft)
# gathering the absolute values for all values in our audio_stft
spectrogram = np.abs(audio_stft)
# Converting the amplitude to decibels
log_spectro = librosa.amplitude_to_db(spectrogram)
# Plotting the short-time Fourier Transformation
plt.figure(figsize=(20, 5))

# Using librosa.display.specshow() to create our spectrogram
librosa.display.specshow(log_spectro, sr=sr, x_axis='time', y_axis='hz',
hop_length=hop_length, cmap='magma')
plt.colorbar(label='Decibels')
plt.title('Spectrogram (dB)', fontdict=dict(size=18))
plt.xlabel('Time', fontdict=dict(size=15))
plt.ylabel('Frequency', fontdict=dict(size=15))
plt.show()



mel_spectrogram = librosa.feature.melspectrogram(y=y, sr=sr, n_fft=n_fft, hop_length=
hop_length, fmax=8000)
log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)
librosa.display.specshow(log_mel_spectrogram, x_axis="time", y_axis="mel", sr=sr)



# Prueba
# AUDIO REAL
path_t = "./Data/ASVspoof2021_LA_eval/flac/LA_E_4495011.flac"

y, sr = librosa.load(path_t, sr=sample_rate)

ipd.Audio(y, rate=sr)

plt.figure(figsize=(20, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveplot', fontdict=dict(size=18))
plt.xlabel('Time', fontdict=dict(size=15))
```

```
plt.ylabel('Amplitude', fontdict=dict(size=15))
plt.show()



# Preprocesado audio

y = TrimAudio(y)
y = CropOrPad(y, audio_len)
y = TimeShift(y)
#y = GaussianNoise(y)
y = y.numpy()

plt.figure(figsize=(20, 5))
librosa.display.waveshow(y, sr=sr)
plt.title('Waveplot', fontdict=dict(size=18))
plt.xlabel('Time', fontdict=dict(size=15))
plt.ylabel('Amplitude', fontdict=dict(size=15))
plt.show()



# Creating a Discrete-Fourier Transform with our FFT algorithm
fast_fourier_transf = np.fft.fft(y)
# Magnitudes indicate the contribution of each frequency
magnitude = np.abs(fast_fourier_transf)
# mapping the magnitude to the relative frequency bins
frequency = np.linspace(0, sr, len(magnitude))
# We only need the first half of the magnitude and frequency
left_mag = magnitude[:int(len(magnitude)/2)]
left_freq = frequency[:int(len(frequency)/2)]
plt.plot(left_freq, left_mag)
plt.title('Discrete-Fourier Transform', fontdict=dict(size=15))
plt.xlabel('Frequency', fontdict=dict(size=12))
plt.ylabel('Magnitude', fontdict=dict(size=12))
plt.show()



# Short-time Fourier Transformation on our audio data
audio_stft = librosa.core.stft(y, hop_length=hop_length, n_fft=n_fft)
# gathering the absolute values for all values in our audio_stft
spectrogram = np.abs(audio_stft)
# Plotting the short-time Fourier Transformation
plt.figure(figsize=(20, 5))
# Using librosa.display.specshow() to create our spectrogram
librosa.display.specshow(spectrogram, sr=sr, x_axis='time',        y_axis='hz', hop_l
ength=hop_length)
plt.colorbar(label='Amplitude')
plt.title('Spectrogram (amp)', fontdict=dict(size=18))
plt.xlabel('Time', fontdict=dict(size=15))
plt.ylabel('Frequency', fontdict=dict(size=15))
plt.show()



# Short-time Fourier Transformation on our audio data
audio_stft = librosa.core.stft(y, hop_length=hop_length, n_fft=n_fft)
# gathering the absolute values for all values in our audio_stft
spectrogram = np.abs(audio_stft)
# Converting the amplitude to decibels
log_spectro = librosa.amplitude_to_db(spectrogram)
# Plotting the short-time Fourier Transformation
plt.figure(figsize=(20, 5))
# Using librosa.display.specshow() to create our spectrogram
librosa.display.specshow(log_spectro, sr=sr, x_axis='time', y_axis='hz', hop_length=h
op_length, cmap='magma')
```

```
plt.colorbar(label='Decibels')
plt.title('Spectrogram (dB)', fontdict=dict(size=18))
plt.xlabel('Time', fontdict=dict(size=15))
plt.ylabel('Frequency', fontdict=dict(size=15))
plt.show()


mel_spectrogram = librosa.feature.melspectrogram(y=y, sr=sr, n_fft=n_fft, hop_length=
hop_length, fmax=8000)
log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)
librosa.display.specshow(log_mel_spectrogram, x_axis="time", y_axis="mel", sr=sr)



log_mel_spectrogram.shape

(128, 481)



# Division entre train y test

X=muestra.drop('class',axis=1)
y=muestra['class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_stat
e=123, stratify=y)
X_train.shape, X_test.shape, y_train.shape, y_test.shape



def generate_mel_spectrograms(df):

    # df: dataframe con el que se trabaja

    features = []
    labels = []
    df = df.reset_index()

    for i in range(len(df)):
        path = df.iloc[i]['filepath']
        class_id = df.iloc[i]['class']
        y, sr = librosa.load(path, sr=sample_rate)
        y = Normalize(y)

        print(path)

        # Preprocesado
        y = TrimAudio(y)
        y = CropOrPad(y, audio_len)
        y = TimeShift(y)
        #y = GaussianNoise(y) # Opcion a introducir ruido gaussiano
        y = y.numpy()

        # Espectrograma de Mel
        mel_spectrogram = librosa.feature.melspectrogram(y=y, sr=sr, n_fft=n_fft, hop
_length=hop_length, fmax=8000)
        log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)
        log_mel_spectrogram = np.reshape(log_mel_spectrogram, (128, 481, 1))

        features.append(log_mel_spectrogram[np.newaxis,...])
        labels.append(class_id)

    features, labels = np.concatenate(features, axis=0), np.array(labels)

    # features: array de espectrogramas extraídos
```

```
    # labels: array de clases

    return features, labels

train_features, train_labels = generate_mel_spectrograms(pd.concat([X_train,y_train],
axis=1))

test_features, test_labels = generate_mel_spectrograms(pd.concat([X_test,y_test],axis
=1))



# One hot encoding para la clase

# Create a dictionary to map the options to numerical values
mapping = {'spoof': 0, 'bonafide': 1}

# Use list comprehension to encode the binary class
train_labels_num = [mapping[option] for option in train_labels]
train_labels_encoded = to_categorical(train_labels_num, num_classes = 2)

test_labels_num = [mapping[option] for option in test_labels]
test_labels_encoded = to_categorical(test_labels_num, num_classes = 2)



# Modelo 1

model1 = Sequential()

model1.add(Input(shape=(128, 481,1)))
model1.add(Conv2D(32, kernel_size=(3, 3), activation='relu',kernel_regularizer=regula
rizers.l2(l=0.01)))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.4))
model1.add(Conv2D(64, kernel_size=(3, 3), activation='relu',kernel_regularizer=regula
rizers.l2(l=0.01)))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.4))
model1.add(Flatten())
model1.add(Dense(128, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Dense(2, activation='softmax'))
model1.summary()

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

predicted_class_indices=np.argmax(pred,axis=1)
predicted_class_indices
```

```
test_labels_array = np.array(test_labels_num, dtype='int64')
test_labels_array

report = classification_report(test_labels_array, predicted_class_indices)
print("Reporte de Clasificación:")
print(report)

cm = confusion_matrix(test_labels_array, predicted_class_indices)
print("Matriz de Confusión:")
print(cm)



# Crear la matriz de confusión utilizando la función heatmap de seaborn
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d", cbar=False, square=True, ax=ax)

# Etiquetas de los ejes
ax.set_xlabel("Predicción")
ax.set_ylabel("Valor Real")

# Etiquetas de los ticks en los ejes x e y
ax.set_xticklabels(["0 (sintético)", "1 (real)"])
ax.set_yticklabels(["0 (sintético)", "1 (real)"])

# Título del gráfico
ax.set_title("Matriz de Confusión")

# Mostrar la figura
plt.show()



# Modelo 2

model1 = Sequential()

model1.add(Input(shape=(128, 481,1)))
model1.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.4))
model1.add(Flatten())
model1.add(Dense(128, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Dense(2, activation='softmax'))
model1.summary()

model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model1.fit(train_features, train_labels_encoded, epochs=15, validation_spli
t=0.2)



# MODELOS PRE-ENTRENADOS



img_input = Input(shape=(128,481,1))
img_conc = Concatenate()([img_input, img_input, img_input])



# VGG 16

base_model = VGG16(input_tensor=img_conc, # Shape of our images
include_top = False, # Leave out the last fully connected layer
weights = 'imagenet')
```

```
for layer in base_model.layers:
    layer.trainable = False

x = layers.Flatten()(base_model.output)

# Add a fully connected layer with 512 hidden units and ReLU activation
x = layers.Dense(512, activation='relu')(x)

# Add a dropout rate of 0.5
x = layers.Dropout(0.5)(x)

# Add a final sigmoid layer with 1 node for classification output
x = layers.Dense(2, activation='softmax')(x)

VGG_16 = tf.keras.models.Model(base_model.input, x)
VGG_16.summary()

VGG_16.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = VGG_16.fit(train_features, train_labels_encoded, epochs=5, validation_split
=0.15)



# VGG 19

base_model_19 = VGG19(input_tensor=img_conc, # Shape of our images
include_top = False, # Leave out the last fully connected layer
weights = 'imagenet')

for layer in base_model_19.layers:
    layer.trainable = False

VGG_19 = tf.keras.models.Model(base_model_19.input, x)
VGG_19.summary()

VGG_19.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = VGG_19.fit(train_features, train_labels_encoded, epochs=5, validation_split
=0.15)



# Inception

base_model = InceptionV3(input_tensor=img_conc, include_top = False, weights = 'image
net')
for layer in base_model.layers:
    layer.trainable = False

x = layers.Flatten()(base_model.output)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)

# Add a final sigmoid layer with 1 node for classification output
x = layers.Dense(2, activation='softmax')(x)

inception = tf.keras.models.Model(base_model.input, x)
inception.summary()

inception.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = inception.fit(train_features, train_labels_encoded, epochs=5, validation_sp
lit=0.15)



# ResNet 50
base_model = ResNet50(input_tensor=img_conc, include_top=False, weights="imagenet")
```

```
for layer in base_model.layers:
    layer.trainable = False

restnet = tf.keras.models.Model(base_model.input, x)
restnet.summary()

restnet.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = restnet.fit(train_features, train_labels_encoded, epochs=5, validation_spli
t=0.15)
```