UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Detección de actividad del habla en vídeos

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

AUTOR/A: Acosta Triana, José Miguel

Tutor/a: Martínez Hinarejos, Carlos David

Director/a Experimental: GIMENO GOMEZ, DAVID

CURSO ACADÉMICO: 2022/2023

Agradecimientos

Deseo expresar mi más sincero agradecimiento a mi tutor, Carlos David, y a mi director experimental, David, por su incesante apoyo y orientación a lo largo de este proyecto. Su dedicación y esfuerzo han sido fundamentales para el éxito de este trabajo.

Asimismo, quiero agradecer a ValgrAI – Valencian Graduate School and Research Network for Artificial Intelligence, asociación a la que estoy adscrito, y a la Generalitat Valenciana por proporcionarme una ayuda económica que me ha permitido centrarme en mis estudios de máster.

Resumen

La detección de actividad del habla en vídeos consiste en identificar el rostro de la persona que está hablando en cada momento de la escena. Este desafío tiene diversas aplicaciones, como pueden ser el enfoque automático en esa persona, la detección de falsificaciones de voz generadas mediante el uso de técnicas de aprendizaje profundo (*DeepFakes*) y la recopilación selectiva de datos para otras tareas, como el entrenamiento de sistemas de lectura de labios automáticos.

En este trabajo, se aborda el problema en dos pasos: la detección de rostros en las imágenes de vídeo y la asociación de los rostros detectados con su correspondiente audio. Ambas etapas se basan en técnicas de aprendizaje automático, siguiendo el proceso estándar de recopilación y etiquetado de datos, selección y entrenamiento de modelos y su posterior evaluación.

El objetivo final del proyecto consiste en facilitar y acelerar el proceso de la anotación de datos para la estimación de sistemas enfocados al reconocimiento del habla audiovisual. Por ello, se ha desarrollado una herramienta capaz de identificar en cada vídeo qué persona está hablando en función del audio correspondiente, recortar las escenas seleccionadas y ofrecérselas al anotador para su posterior supervisión.

Palabras clave— Detección del Habla, Aprendizaje Automático, Visión por Computador, Detección de Caras, Anotación de Datos, *Deep Learning*

Abstract

Activity speech detection in videos consists of identifying the face of the person who is speaking at each moment of the scene. This challenge has various applications, such as automatic focusing on the person, detection of voice impersonation generated by using deep learning techniques (DeepFakes), and selective data collection for other tasks, such as training automatic lip-reading systems.

In this project, the problem is approached in two steps: face detection in video frames and associating the detected faces with its corresponding audio. Both stages rely on machine learning techniques, following the standard process of data collection and labeling, model selection and training, and subsequent evaluation.

The ultimate goal of the project is to facilitate and speed up the process of data annotation for the estimation of systems focused on audiovisual speech recognition. For this purpose, we have developed a tool capable of identifying which person is speaking in each video based on the corresponding audio, cropping the selected scenes, and offering them to the annotator for further supervision.

Keywords— Speech Detection, Machine Learning, Computer Vision, Face Detection, Data Annotation, Deep Learning

Resum

La detecció de l'activitat de parla en vídeos consisteix a identificar el rostre de la persona que està parlant en cada moment de l'escena. Aquest desafiament té diverses aplicacions, com poden ser l'enfocament automàtic en aquesta persona, la detecció de falsificacions de veu generades mitjançant l'ús de tècniques d'aprenentatge profund (*DeepFakes*) i la recopilació selectiva de dades per a altres tasques, com l'entrenament de sistemes de lectura de llavis automàtics.

En aquest treball, s'aborda el problema en dos passos: la detecció de rostres a les imatges de vídeo i l'associació dels rostres detectats amb el seu corresponent àudio. Totes dues etapes es basen en tècniques d'aprenentatge automàtic, seguint el procés estàndard de recopilació i etiquetatge de dades, selecció i entrenament de models i la seva posterior avaluació.

L'objectiu final del projecte consisteix a facilitar i accelerar el procés de l'anotació de dades per a l'estimació de sistemes enfocats al reconeixement de la parla audiovisual. Per això, s'ha desenvolupat una eina capaç d'identificar en cada vídeo quina persona està parlant en funció de l'àudio corresponent, retallar les escenes seleccionades i oferir-les a l'anotador per a la seva posterior supervisió.

Paraules Clau— Detecció de la Parla, Aprenentatge Automàtic, Visió per Computador, Detecció de Cares, Anotació de Dades, Aprenentatge Profund.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1 Motivación

Actualmente, en la era de la información y la tecnología, el análisis de audio y vídeo se ha convertido en un campo de investigación esencial, cuyas aplicaciones cubren una gran variedad de áreas, como son la medicina [1, 2], el reconocimiento del habla [3, 4] o la conducción de vehículos autónomos [5, 6]. Una de las tareas que está recibiendo un creciente interés en este campo es la detección de hablantes en vídeos [7, 8, 9], constituyendo un problema abierto de investigación al presentar numerosos desafíos y oportunidades.

La detección de los hablantes en vídeos tiene una gran importancia en múltiples contextos. Por ejemplo, en la industria del entretenimiento y los medios de comunicación, es fundamental para indexar y organizar grandes volúmenes de contenido audiovisual [10]. Esto permite una mejor clasificación y búsqueda de vídeos, facilitando la recuperación de información relevante según el locutor que pueda mejorar la experiencia del usuario.

El enfoque automático de locutores puede ser un área donde la detección del hablante juegue un papel fundamental, ya que permitiría hacer zoom a la persona que esté hablando en un vídeo. Esta funcionalidad resulta especialmente útil en programas de noticias como pueden ser telediarios, donde es crucial tener una visión nítida del locutor que pueda beneficiar a las personas con dificultades auditivas [11].

Además, en el campo de la seguridad, la identificación de hablantes puede ser considerada de vital importancia, puesto que ayuda en la detección de falsificaciones de voz generadas mediante técnicas de aprendizaje profundo (*Deepfakes*) [12], puesto que en gran parte de los casos los movimientos de la boca no se corresponden con los que debería realizar el locutor. Es de gran importancia contar con herramientas que asistan en esta tarea, ya que se ha convertido en un desafío cada vez mayor. Una herramienta capaz de detectar automáticamente estas falsificaciones puede ayudar a prevenir

la difusión de información falsa y proteger la integridad de los sistemas de comunicación.

Otro uso de las tecnologías de detección del hablante es la interacción humano-robot [13]. A medida que los robots se vuelven más comunes en nuestras vidas, es esencial que sean capaces de reconocer y adaptarse a las características individuales de los hablantes, puesto que mediante el reconocimiento del interlocutor se puede llegar a alcanzar una comunicación más natural y personalizada. De este modo, se mejoraría la experiencia de interacción con los robots en diversos entornos, como el hogar, la educación y la atención médica.

En este contexto, el desarrollo de una herramienta que permita detectar el hablante en vídeos y generar muestras para crear bases de datos [14, 15] es una contribución significativa al campo de las tecnologías del habla audiovisuales. Esta herramienta no solo facilitará la identificación precisa de hablantes en diversas aplicaciones, sino que también supondrá un gran avance para entrenar modelos enfocados a otro tipo de tareas, como puede ser la lectura de labios [16, 17].

No obstante, al abordar esta tarea debemos ser conscientes de los desafíos que presenta, ya que surgen posibles dificultades como voces en off, que no deben ser confundidas en caso de que alguna persona del vídeo esté moviendo la boca. También se pueden dar casos de personas hablando en otro idioma, donde el audio proporcionado es una traducción; lógicamente, en este caso no se corresponden el audio y el vídeo, y no debe clasificarse como si el locutor estuviese hablando. Adicionalmente, pueden haber personas de fondo hablando pero sin escucharse en el audio proporcionado, donde en algunos conjuntos de datos de entrenamiento se marcan estos casos de forma explícita para aprender a modelarlos [18]. Como en cualquier otro problema de clasificación, si la calidad del audio o del vídeo de entrada es mala, se complicará substancialmente la detección de actividad del habla.

1.2 Objetivos

En este trabajo contamos con dos objetivos principales. En primer lugar deseamos crear un sistema automático enfocado a la detección del habla para el español, que tome vídeos como entrada y sea capaz de detectar el habla en los mismos, transcribir los segmentos donde detecte locutores y guardar todos los datos respectivos a esos segmentos. Como segundo objetivo, deseamos una herramienta final que facilite y acelere el proceso de anotación de datos audiovisuales que puedan ser de gran interés para el campo de las tecnologías del habla, ahorrando tiempo y costes económicos a la hora de anotar datos audiovisuales.

Esta herramienta debe ser visualmente agradable, intuitiva, sencilla de utilizar y permitir correcciones en la transcripción de audio generada au-

tomáticamente. Además, debe ser rápida para que el usuario no tenga que esperar por la visualización de las muestras generadas y acelerar el proceso de creación de la base de datos. Adicionalmente, esta herramienta debe poderse ejecutar tanto en sistemas operativos Windows como Linux, para poder ofrecer un amplio nivel de compatibilidad. El usuario debe poder también suministrar su propio modelo de detección del hablante si así lo desea, ya que puede funcionar mejor para otro dominio u otro idioma que quiera emplear.

Para poder llevar a cabo este proyecto es de importancia contar con sub-objetivos que nos ayuden a lograr los objetivos finales, particionando el trabajo en metas.

El primer sub-objetivo de este trabajo es emplear un conjunto de datos existente que sea representativo y adecuado para el problema a solucionar, procesarlo para adecuarse al uso con modelos de detección del hablante y acelerar la carga de los datos posteriormente.

Tras esto es fundamental seleccionar un modelo de detección del hablante adecuado para poder entrenar o realizar *fine-tuning* sobre él, que sea capaz de obtener un buen rendimiento con duraciones de muestra cortas y posea robustez ante diferentes circunstancias y hablantes. Preferiblemente, debe contar con un tiempo de inferencia rápido.

Es crucial además elegir modelos de detección de caras y reconocimiento del habla aptos para el procesado de los vídeos de entrada. El detector de caras debe ser preciso incluso en vídeos con baja resolución y tener un tiempo de inferencia bajo, y el modelo de reconocimiento de habla tiene que ser capaz de lograr unos buenos resultados transcribiendo el lenguaje español, independientemente de qué acento tenga el hablante o la existencia de cierta cantidad de ruido de fondo, puesto que es imposible garantizar la homogeneidad en los vídeos de entrada.

1.3 Estructura

En lo que respecta a la estructura del proyecto, se ha dividido en diferentes capítulos, de modo que cada uno cubre una temática distinta sobre el presente trabajo.

En este primer capítulo se ha introducido el proyecto, señalando los motivos por los que se ha llevado a cabo. De igual forma, se han explicado los objetivos propuestos que se pretenden alcanzar mediante el desarrollo de la herramienta final y lo que se espera lograr con ella.

Tras ello, en el **Capítulo 2** se presentan los conjuntos de datos principales que se utilizan tanto para entrenar como para evaluar sistemas para esta tarea, y las arquitecturas más utilizadas actualmente así como modelos específicos del estado del arte. Además, se mencionan y explican algunas de

las métricas más comunes a la hora de evaluar un sistema de detección del habla.

Acto seguido, en el **Capítulo 3** se comentan en detalle las principales herramientas tecnológicas de las que hemos hecho uso para el desarrollo de la herramienta, incluyendo los motivos que nos han llevado a escogerlas y su funcionamiento.

En el **Capítulo 4**, explicamos el conjunto de datos empleado para la tarea, cómo se compone, de qué forma se ha preprocesado y qué procedimiento se ha seguido para generar muestras adecuadas para la tarea.

Posteriormente, el **Capítulo 5** detalla el proceso de experimentación llevado a cabo para seleccionar los mejores parámetros para el modelo, los resultados obtenidos y los métodos utilizados para mejorar el rendimiento final de la herramienta ante escenarios más realistas.

En penúltimo lugar, el **Capítulo 6** describe toda la estructura de la herramienta, así como las partes que la componen, cómo funcionan, el proceso por el que pasan los datos para ser analizados, y qué opciones se le brindan al usuario a la hora de personalizar su experiencia de uso.

Para finalizar, el **Capítulo 7** cubre las conclusiones del proyecto, donde se resumen los puntos principales de este documento y se indican los resultados obtenidos tras la finalización del desarrollo de la herramienta. Adicionalmente, se explican los trabajos futuros o ampliaciones de la herramienta que no se han podido llevar a cabo debido a limitaciones temporales.

Capítulo 2

Estado del arte

Antes de adentrarse en la herramienta creada y su funcionamiento, es importante conocer las tecnologías del estado del arte que permiten que esta clase de desarrollos sean posibles, incluyendo conjuntos de datos con los que entrenar y evaluar los modelos, qué arquitecturas y modelos específicos se utilizan actualmente o incluso cómo se evalúan estos modelos al entrenarlos para asegurar robustez y buenos resultados.

2.1 Conjuntos de datos

2.1.1 AVA-ActiveSpeaker

El conjunto de datos AVA-ActiveSpeaker [18] es uno de los más utilizados en el entrenamiento y evaluación de modelos centrados en la tarea de *active speaker detection* actualmente. Ha sido creado por Google y cuenta con alrededor de 3.65 millones de fotogramas de caras etiquetados a mano. De este modo, la base de datos ofrece, aproximadamente, unas 38.5 horas de vídeo. Concretamente, cada uno de los vídeos tiene una duración de entre 15 y 30 minutos en términos medios. Cabe destacar que estos vídeos pertenecen a fragmentos de películas de distintos géneros e idiomas, englobando un total de 188 películas distintas. Este conjunto de datos es una versión modificada del conjunto AVA [19], usado para comprender acciones y gestos humanos.

Se dispone de distintos archivos en formato CSV, cada uno para una partición distinta del conjunto, lo que permitirá comparaciones justas entre modelos. En cada archivo se indica para cada fotograma: el vídeo al que pertenece, el instante de tiempo en el que se encuentra el fotograma en segundos, las coordenadas para las *bounding boxes* de la cara detectada, una etiqueta identificadora única para cada segmento de esa cara y, por último, una etiqueta que indica el tipo de muestra entre las tres clases posibles (no habla, habla pero no es audible, habla y es audible). La distribución de las

muestras dentro del conjunto de datos se expresa en la Tabla 2.1.

Clase	Tiempo	Nº segmentos	Duración media
No habla	28.1 horas	58171	1.74 segundos
Habla (audible)	9.5 horas	30623	1.11 segundos
Habla (no audible)	0.4 horas	1547	0.83 segundos

Tabla 2.1: Distribución de las distintas clases de muestras del conjunto de datos AVA-ActiveSpeaker.

Como se puede observar, la base de datos se encuentra considerablemente desequilibrada entre clases. Por tanto, muchos modelos hacen uso de este conjunto para entrenar pero evalúan adicionalmente con conjuntos distintos como pueden ser Columbia ASD [20] o WASD [21].

2.1.2 Columbia ASD

El conjunto de datos Columbia ASD [20] se utiliza en una gran cantidad de trabajos del estado del arte para evaluar el rendimiento en casos reales. Se trata de un vídeo de 87 minutos manualmente etiquetado. En él podemos ver un panel de discusión de la Universidad de Columbia, donde hay siete hablantes distintos y donde puede haber hasta dos o tres posibles hablantes visibles en cualquier momento dado. Debido al caso real que presenta, así como al desafío de su baja resolución (1280×720 píxeles), y ha ganado gran popularidad en el campo de la detección del hablante.

La baja resolución supone un desafío adicional, ya que, como otros estudios han demostrado [22, 23], la lectura de labios se ve afectada por ello. Por lo tanto, esta base de datos permite evaluar la robustez de los modelos frente a este tipo de escenarios más realistas.

2.2 Modelos de detección del hablante

En esta sección se describen los modelos de detección automática del habla SPELL y Light-ASD. El modelo abordado en nuestro proyecto, TalkNet-ASD, se describe con detalle en el Capítulo 3.

Actualmente, el estado del arte de los modelos de detección de hablante hacen uso de una o varias de las siguientes arquitecturas o mecanismos:

- *Conformer* [24]: Es una fusión entre el uso de *transformers* [25] y redes neuronales convolucionales (CNN, por sus siglas en inglés) [26, 27]. Se ha utilizado sobre todo en el campo de procesamiento del lenguaje natural y reconocimiento del habla. Si bien los *transformers* son capaces de modelar con precisión las relaciones a largo plazo entre los

datos de entrada, las redes neuronales convolucionales pueden captar patrones y características locales. Gran parte del éxito de este enfoque se base en aprovechar las fortalezas de ambas arquitecturas para mejorar el rendimiento de los modelos. Los *conformers* representan el estado del arte actual en varias tareas de aprendizaje automático [28, 29].

- Redes neuronales gráficas (GNN, por sus siglas en inglés) [30]: Son un tipo de arquitectura de redes neuronales que se utilizan para modelar datos estructurados en forma de grafos. A diferencia de las redes neuronales convencionales que están diseñadas para datos tabulares o secuenciales, las redes neuronales gráficas pueden trabajar directamente con datos que poseen relaciones complejas y no lineales entre sí. Normalmente, en la detección del hablante [31] primero se convierte la información de interés extraída de los vídeos a forma de grafo con sus nodos y aristas, y se tienen en cuenta las relaciones temporales para clasificar los nodos. La representación de la información en forma de nodos y aristas no es fija, y depende de la implementación llevada a cabo. En la mayoría de casos, esta conversión de los datos a forma de grafo se realiza como preprocesado de los datos, ahorrando tiempo a la hora de entrenar el modelo.

- Mecanismos de auto-atención [25]: Estos mecanismos constituyen el principal componente de la arquitectura *transformer* [25]. Su éxito radica en su gran capacidad a la hora de capturar las relaciones y las dependencias entre diferentes elementos de una secuencia. Se basan en calcular las interacciones entre todos los elementos de la secuencia de entrada y hacer uso de ellas para determinar la importancia relativa de cada elemento dentro de la secuencia. De esta forma, el modelo logra centrar su atención en aquellos elementos de la secuencia que presentan una mayor importancia de cara a la tarea abordada.

2.2.1 SPELL

SPELL [31] es un modelo presentado en 2022 y actualmente obtiene los mejores resultados en el conjunto de validación de la base de datos AVA-ActiveSpeaker. Concretamente, se obtiene alrededor de un 94.2 % mAP. SPELL hace uso de la arquitectura ResNet [27] y redes neuronales gráficas (GNNs). Además, cabe destacar que los vídeos son transformados en un grafo multimodal, donde el audio, los fotogramas y los diferentes hablantes se integran a lo largo del tiempo, pudiendo modelar así relaciones complejas que puedan existir entre los datos de entrada.

En primer lugar, esta arquitectura realiza un preproceso sobre el vídeo. De este modo, se define un grafo que no solo tiene en cuenta las personas existentes en cada fotograma, sino también su relación a lo largo del tiempo.

Con el objetivo de codificar las características audiovisuales, se utiliza una red ResNet 2D *two-stream* [32], analizando así la información tanto espacial como temporal del vídeo. Adicionalmente, se le proporciona información sobre la localización en el vídeo de cada cara, puesto que presenta información discriminativa a la hora de clasificar. Se obtienen de este proceso tres grafos distintos, uno dirigido con las aristas conectando a la misma persona hacia delante en el eje temporal, otro dirigido conectando hacia atrás en el eje temporal y por último un grafo no dirigido. Mediante la creación de múltiples grafos que relacionan de forma distinta los datos, se pueden modelar mejor las características audiovisuales. Se puede observar un esquema indicativo de la entrada y el resultado en la Figura 2.1.

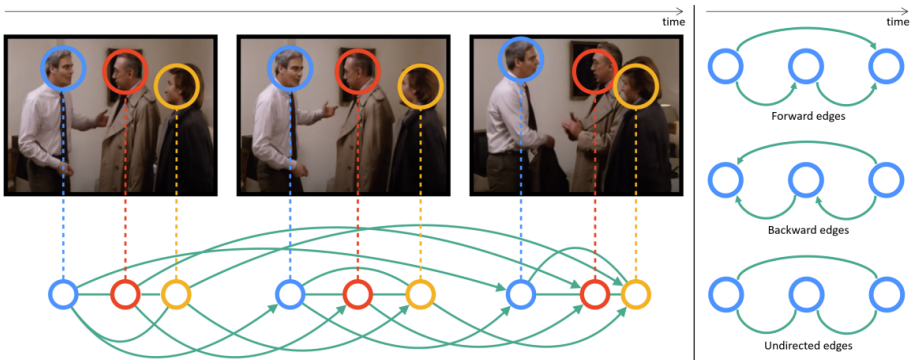


Figura 2.1: Creación del grafo con SPELL [31].

A la hora de clasificar el vídeo se hará uso de una GNN bidireccional convolucional [33, 34] basada en tres módulos distintos, a la que se le pasarán los tres grafos obtenidos anteriormente. Cada módulo tiene tres capas, donde los parámetros de la segunda capa lo comparten los tres módulos. La primera capa se basa en las conocidas como *edge-convolutions* [35], mientras y las dos últimas se basan en *SAGE-convolutions* [36].

La gran ventaja que tiene este modelo es que, al tomar en cuenta los múltiples individuos que puedan aparecer en una misma escena para construir el grafo, elimina en gran parte el problema de repetir el proceso de clasificación con cada persona de forma independiente cuando solo una de ellas estaría hablando. En otras palabras, otros modelos [7, 37], al no tener en cuenta esta información, pueden tender a proporcionar más falsos positivos si otra persona en la escena está moviendo la boca o llevando a cabo una acción similar, ya que no tienen en cuenta el contexto completo.

2.2.2 Light-ASD

Light-ASD [7] es un modelo presentado en 2023 capaz de conseguir resultados comparables al actual estado del arte con una cantidad de parámetros mínima. Concretamente, esta arquitectura cuenta únicamente con un millón

de parámetros. Se hace uso de convoluciones *depthwise* 2D y 3D "separadas" [38] para extraer características audiovisuales. Una vez extraídas estas características, se aplican unidades recurrentes cerradas (GRU, por sus siglas en inglés) [39] para modelar las relaciones temporales que puedan existir entre ellas. El modelo logra un mAP (94.1 %) casi idéntico al del modelo SPELL (94.2 %). Sin embargo, el modelo Light-ASD llega a ser cuatro veces más rápido.

Más en detalle, tal y como refleja la Figura 2.2, el modelo cuenta con un *encoder* para el vídeo y otro para el audio con el objetivo de extraer las características más importantes de ambas modalidades. Estos *encoders* fueron construidos de la siguiente manera:

- El *encoder* de vídeo consta de tres bloques visuales, estando todos ellos unidos entre sí mediante capas de *max pooling*. Dentro de cada bloque se realizan convoluciones 3D separadas [38] en distintas convoluciones 2D y 1D para mejorar el tiempo de ejecución y minimizar el número de parámetros. Para ello, en primer lugar, se extraen distintas características con diferentes tamaños de filtro 2D, y es entonces cuando, al final del bloque, se combinan las mismas haciendo uso de una convolución 1D.

- El *encoder* de audio funciona de manera similar, estando compuesto por tres bloques de audio. En ellos, se usan convoluciones 2D partidas en diferentes convoluciones 1D por dos caminos separados, uniéndose con otra convolución final a la salida del bloque.

Las salidas de ambos *encoders* se unifican mediante la suma elemento a elemento de las características obtenidas. Posteriormente, esta representación audiovisual es procesada por modelos GRU bidireccionales que modelarán el contexto temporal de las características obtenidas. Finalmente se clasifica fotograma a fotograma si la persona en cuestión está hablando o no mediante una capa densa. En la Figura 2.2, se encuentra una visualización de la arquitectura general del modelo.

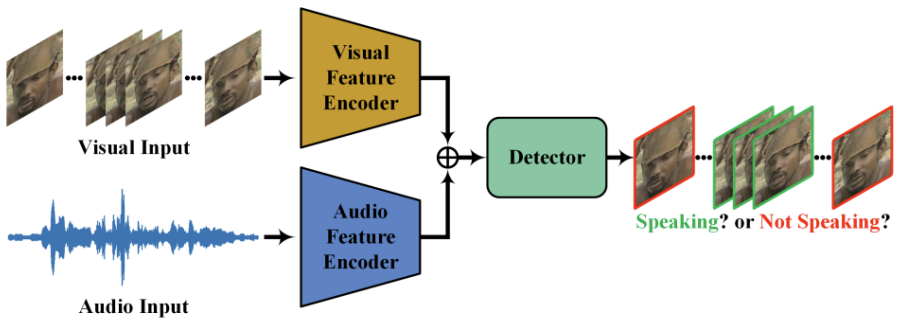


Figura 2.2: Arquitectura de Light-ASD [7].

2.3 Evaluación de modelos de detección del hablante

A la hora de entrenar y poner en marcha un modelo de aprendizaje automático, es de interés conocer la correcta metodología de evaluación del mismo. Una buena evaluación puede ayudar a encontrar fallos, además de guiar hacia posibles modificaciones y ajustes para mejorar el rendimiento del modelo.

Antes de mencionar las métricas principales de evaluación de modelos de detección del habla, conviene introducir algunas otras métricas también muy relevantes en el campo de la clasificación automática:

- **Verdaderos Positivos (TP)**: Son los casos en los que el modelo predice correctamente la clase positiva. Es decir, los casos en los que el modelo acierta al afirmar que un ejemplo pertenece a la clase que estamos interesados en identificar.

- **Falsos Positivos (FP)**: Estos son los casos en los que el modelo predice incorrectamente la clase positiva. En otras palabras, el modelo hizo una afirmación incorrecta al decir que un ejemplo pertenece a la clase que estamos interesados en identificar a pesar de no ser el caso.

- **Falsos Negativos (FN)**: Los falsos negativos son los casos en los que el modelo predice incorrectamente la clase negativa. Esto significa que el modelo no pudo detectar ejemplos que realmente pertenecen a la clase positiva.

- **Verdaderos Negativos (TN)**: Son los casos en los que el modelo predice correctamente la clase negativa. Esto significa que el modelo acertó al decir que un ejemplo no pertenece a la clase que estamos interesados en identificar.

- **Precisión**: La precisión mide la proporción de predicciones positivas hechas por el modelo que fueron correctas. Un elevado valor de precisión significa que el modelo tiene menos falsos positivos. Su cálculo se define en la ecuación 2.1.

$$Precisión = \frac{TP}{(TP + FP)} \tag{2.1}$$

- ***Recall***: El *recall* mide la proporción de muestras positivas reales que el modelo ha identificado correctamente. Un valor elevado indica que el modelo tiene menos falsos negativos. Su cálculo se define en la ecuación 2.2.

$$Recall = \frac{TP}{(TP + FN)} \tag{2.2}$$

A continuación se mencionan las métricas más ampliamente utilizadas para evaluar sistemas de detección del hablante, ordenadas de menor a mayor complejidad:

- **Tasa de acierto (*accuracy*):** Es una de las métricas más sencillas que existen pero también de las más utilizadas, ya que proporciona una idea general y directamente relacionada con el rendimiento del modelo. Se basa en la relación entre las muestras correctamente predichas por el modelo y las muestras totales. Su cálculo viene dado por la ecuación 2.3.

$$Tasa\ de\ acierto = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{2.3}$$

  Adicionalmente, se suele calcular un intervalo de confianza, que nos proporcionará cierto margen para poder comparar resultados y asegurarnos de que son representativos y no se solapan. Esto es especialmente útil cuando no se cuenta con muchos datos para evaluar, ya que los resultados pueden variar considerablemente incluso dentro de una misma prueba. El intervalo de confianza más utilizado es normalmente el de 95 %, donde en este caso indica que hay un 95 % de probabilidad de que el intervalo que se ha estimado incluya el verdadero valor. La fórmula para su cálculo se encuentra en la ecuación 2.4, donde N es el número de muestras del conjunto con el que se evalúa y $\hat{p}$ es la tasa de acierto obtenida.

$$\hat{p} \pm 1{,}96\sqrt{\frac{\hat{p}(1-\hat{p})}{N}} \tag{2.4}$$

- **AUC (Área bajo la curva ROC):** Métrica que tiene en cuenta las tasas de verdaderos positivos y falsos positivos. Permite la comparación de uno o varios modelos con distintos umbrales de clasificación. Adicionalmente, resulta útil cuando nos enfrentamos a conjuntos de datos desequilibrados o cuando deseamos determinar el umbral óptimo de clasificación [1].

- **mAP (*mean average precision*)**: Métrica que hace uso de la precisión y el *recall*. Se calculan estos dos indicadores para cada clase evaluada y se calcula la media según el método de PASCAL VOC [40].

[1] `https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/`

Capítulo 3

Tecnologías utilizadas

3.1 Dual Shot Face Detector

Dual Shot Face Detector (DSFD) [41] es un modelo de detección de caras del estado del arte que logra una alta precisión al detectar rostros en imágenes.

Concretamente, DSFD fue diseñado para abordar el desafío de detectar caras en una alta variabilidad de escenarios con diferencias en diversos aspectos como la escala, pose e iluminación, entre otros. Tal y como su nombre indica, sigue una estrategia de detección en dos pasos que consiste en un detector de caras aproximado y un detector de caras refinado. Este enfoque de doble paso permite que DSFD pueda manejar caras de diferentes escalas de una forma más efectiva.

Este modelo hace uso de versiones modificadas de las redes neuronales VGG16 [26] y ResNet [27]. Cuenta con tres módulos principales, uno para la detección de caras aproximada, otro para la "mejora" de los mapas de características obtenidos ("Feature Enhance Module") y, finalmente, un módulo para la detección refinada de estas características. El proceso que ejecuta es el siguiente:

- En el primer paso, el detector de rostros aproximado se encarga de predecir cajas candidatas iniciales de caras. Estas cajas candidatas cubren una amplia gama de posibles ubicaciones y escalas de rostros en la imagen. Para ello, se hace uso de *anchor boxes* de un tamaño reducido durante esta fase, tal y como se propuso en el diseño del modelo Faster R-CNN [42].

- El "Feature Enhance Module" en DSFD se encuentra entre la etapa de detección aproximada y la etapa de detección refinada. Su objetivo es mejorar la calidad de los mapas de características generados por la etapa de detección aproximada, ayudando así a obtener mejores resultados en la etapa de detección refinada. Este módulo hace una

multiplicación elemento a elemento del mapa de entrada con el mapa del siguiente nivel de profundidad. El mapa resultante se divide en tres partes, donde cada una pasa por una serie de entre una y tres convoluciones dilatadas [43], siendo posteriormente concatenados los resultados a un mapa de características del mismo tamaño que el de entrada. De esta manera, se aprovecha la información contextual y las relaciones espaciales entre las características, resaltando y mejorando aquella información que pueda ser más relevante para la detección de caras.

- El detector de rostros refinado utiliza las características extraídas de la etapa de detección aproximada para refinar las cajas candidatas, haciendo uso, en este caso, de *anchor boxes* de mayor tamaño para detectar caras.

A la salida del modelo se utilizan tanto los mapas de características originales como los mejorados, haciendo uso de dos capas separadas que se relacionarán mediante la función de pérdida del modelo. En la Figura 3.1 se puede observar una representación gráfica de la arquitectura empleada.

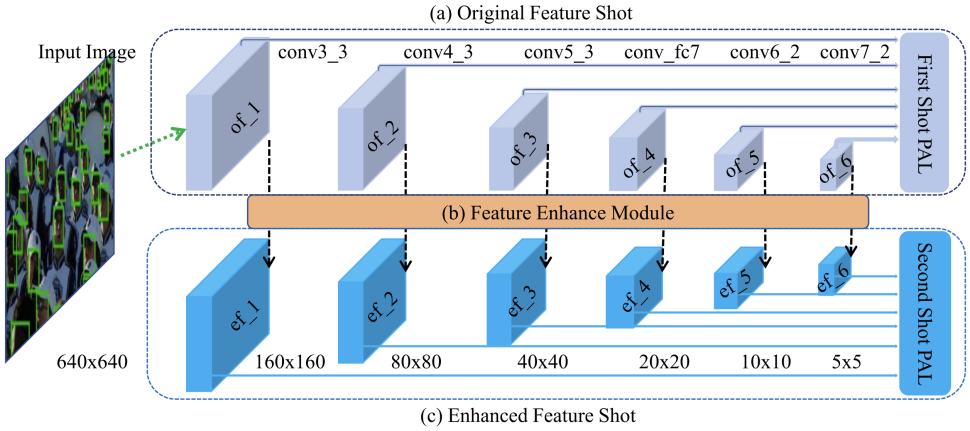


Figura 3.1: Arquitectura de Dual Shot Face Detector (DSFD) [41].

Cabe destacar que, para mejorar el rendimiento del modelo, se emplean técnicas de agrupación de *anchor boxes*. Además, durante el entrenamiento, se hacen uso de técnicas de *data augmentation* para proporcionarle una mayor robustez al modelo. Otro detalle a resaltar del entrenamiento del modelo es que se hacen uso de más *anchor boxes* obtenidas de los mapas de características mejorados que de los mapas de características originales.

En general, DSFD ha logrado un rendimiento excelente en *benchmarks* ampliamente utilizados para la detección de rostros, tal y como puede ser WIDER FACE [44], donde obtiene una tasa de acierto media de 96.0 %, 95.3 % y 90.0 % en sus subconjuntos *easy*, *medium* y *hard*, respectivamente.

Estos resultados demuestran su eficacia para detectar con precisión rostros con diferentes escalas y poses. Una de las razones por las que escogimos este modelo en nuestro proyecto fue porque incluso en la actualidad, cuatro años después de su presentación, se encuentra entre los detectores de caras con mejor rendimiento.

3.2 TalkNet-ASD

Con el objetivo de crear nuestra herramienta, tuvimos que seleccionar un modelo de detección del hablante adecuado, para entrenar con nuestro conjunto de datos descrito en el Capítulo 4. Para ello, hemos elegido un modelo llamado TalkNet-ASD [37]. Comparativamente, TalkNet obtiene un mAP de 92.3 % en el conjunto de validación de la base de datos AVA-ActiveSpeaker [18] mientras que SPELL [31], el mejor modelo actualmente en este conjunto de datos, logra un 94.2 %.

La razón por la que hemos elegido este modelo en vez del modelo SPELL (mencionado en el Capítulo 2) es que, según hemos visto en los experimentos, TalkNet-ASD funciona mejor con contextos audiovisuales cortos, que es lo que nos interesa, ya que en muchas ocasiones los vídeos que vamos a tener en nuestro dominio van a ser con planos de hablantes muy breves. Observamos que existe una diferencia de rendimiento, pero teniendo en cuenta que el conjunto de datos AVA-ActiveSpeaker posee vídeos con duraciones mucho mayores a las que vamos a utilizar nosotros, creemos que esta diferencia de rendimiento puede resultar menos significativa en nuestro caso o incluso invertirse. Además, el uso de SPELL requiere un preproceso muy específico de los datos para convertirlos a grafos, con lo que limitaría considerablemente la posibilidad de intercambiar el modelo de detección del habla utilizado por la herramienta final, Por el contrario, TalkNet-ASD únicamente requiere como entrada el audio y los fotogramas de la cara a analizar, al igual que la mayoría de modelos existentes en esta área, permitiendo así una mayor flexibilidad al usuario final para usar un modelo de detección del hablante más adecuado a la lengua o escenarios que vaya a abordar. No se ha utilizado tampoco Light-ASD [7] puesto que ha surgido muy recientemente (concretamente, durante la realización de este proyecto, cuando ya se había realizado toda la experimentación sobre el modelo).

El modelo TalkNet-ASD se basa principalmente en la arquitectura ResNet [27], así como en el uso de mecanismos de auto-atención [25] para aprender a modelar las características de vídeo y audio.

TalkNet-ASD toma como entrada los fotogramas de caras de la persona a analizar, en resolución 112×112 y escala de grises, además del audio procesado en forma de 13 coeficientes cepstrales de Mel. Este modelo cuenta con dos codificadores, uno para el audio y otro para el vídeo. El codificador de vídeo consta de una primera convolución 3D sobre los datos seguido de una

red ResNet-18; tras ello se pasará su salida por una serie de bloques convolucionales con conexiones residuales; por último, se aplica una convolución unidimensional. El codificador de audio utiliza una ResNet-34 modificada con convoluciones dilatadas para que las características del audio estén alineadas temporalmente con las del vídeo; además, este codificador hace uso de un módulo añadido de *squeeze-and-excitation* [45], que permite modelar explícitamente las dependencias entre canales y buscar los canales más informativos, mejorando la discriminación de características.

Una vez obtenidas las características codificadas tanto de audio como de vídeo (F_a y F_v, respectivamente), éstas serán procesadas por dos módulos de atención cruzada, tal y como se puede observar en la Figura 3.2. El objetivo consistiría en combinar y relacionar las características de cada modalidad, como pueden ser el contenido fonético o ciertos patrones del habla del locutor, además de aprender una representación conjunta audiovisual, interpretando cómo cada una de las entradas (audio y vídeo) influye sobre la otra. Las entradas son el vector *query* Q =(Q_a,Q_v), clave K =(K_a,K_v) y valor V =(V_a,V_v), determinando qué instantes de tiempo (*frames*) son más relevantes de V con respecto a Q y K. Se obtiene como salida resultante $F_{a\to v}$ y $F_{v\to a}$. Las dos redes son iguales, pero una calcula $F_{a\to v}$ y la otra $F_{v\to a}$. Las fórmulas usadas por cada red se muestran en las ecuaciones 3.1 y 3.2, donde d denota la dimensión de Q, K y V.

$$F_{a\to v} = softmax\left(\frac{Q_v K_a^T}{\sqrt{d}}\right) V_a \tag{3.1}$$

$$F_{v\to a} = softmax\left(\frac{Q_a K_v^T}{\sqrt{d}}\right) V_v \tag{3.2}$$

Se pasan las salidas $F_{a\to v}$ y $F_{v\to a}$ de los módulos de atención cruzada por una capa *feed-forward* y se concatenan las salidas de las dos redes, que serán proporcionadas como entrada al módulo de auto-atención. Este módulo presenta un comportamiento similar al de las redes de atención cruzada pero, en este caso, los vectores de *query* (Q_{av}), clave (K_{av}) y valor (V_{av}) harán uso de las características audiovisuales conjuntas obtenidas previamente. El objetivo de este módulo es distinguir los fotogramas de la muestra en los que se habla y en los que no. Finalmente, se usa una capa densa y una función de activación *softmax* para clasificar la salida del módulo de auto-atención, proporcionando una etiqueta por cada fotograma del vídeo. En la Figura 3.2 se observa un esquema general de la arquitectura del modelo TalkNet-ASD.

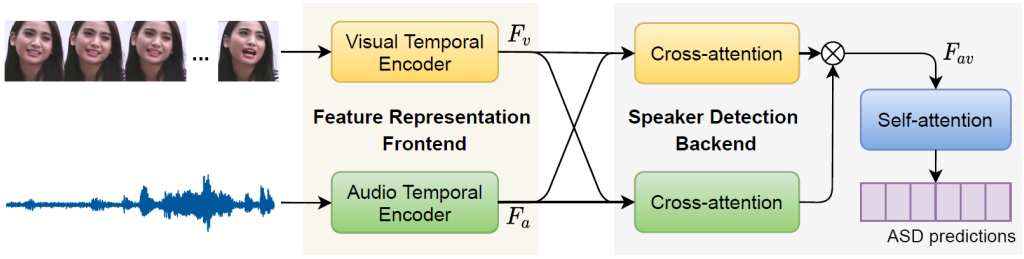


Figura 3.2: Visualización de la arquitectura de Talknet-ASD [37].

Este modelo normalmente no procesa un vídeo entero, sino un fragmento del mismo de cierto tamaño; a este tamaño lo llamaremos "ventana de contexto" a lo largo de este trabajo. Esta ventana o tamaño de contexto se fijará previamente a la hora de recortar los vídeos de entrada y suministrárselos al modelo, puesto que los vídeos a analizar por la herramienta final potencialmente serán de larga duración, incapaces por tanto de ser analizados directamente por el modelo, y deberán ser "troceados" en segmentos breves.

3.3 Whisper

Whisper [3] es un modelo de reconocimiento automático del habla desarrollado por OpenAI. Funciona utilizando técnicas de aprendizaje profundo y se basa en la arquitectura *transformer* para convertir el habla en texto. Es capaz de reconocer habla en múltiples idiomas, además de contar con la capacidad de identificar automáticamente el lenguaje del que proviene el audio de entrada. Es actualmente el modelo de reconocimiento del habla más conocido y uno de los más utilizados, ya no sólo por su sencillez de instalar y ejecutar [1], sino también por los excelentes resultados que proporciona.

El funcionamiento de Whisper se basa en el entrenamiento con grandes conjuntos de datos de voz y texto. Durante el entrenamiento, el modelo aprende a asociar los patrones de audio con las palabras correspondientes. Esto permite que el modelo pueda transcribir el habla en texto de manera precisa y automatizada. Ha sido entrenado con más de 680.000 horas de datos supervisados, multilingües, multitarea y recolectados de la web. Aproximadamente un tercio de los datos de entrenamiento son en un idioma distinto del inglés. Cabe destacar que los mejores resultados de ratio de error por palabra medio (WER) son para el español, con lo que constituye una buena elección para nuestra herramienta, ya que hará uso de un corpus en lengua española.

Whisper utiliza como base una architectura *transformer* [25], que permite utilizar el mecanismo de atención para capturar las relaciones a largo

[1] `https://github.com/openai/whisper`

plazo entre las características acústicas. La atención permite que el modelo se enfoque en partes específicas de la secuencia de entrada y asigne pesos a esas partes en función de su importancia relativa. Primeramente, divide el audio en fragmentos de unos 30 segundos que se irán procesando uno a uno; se calcula la representación del espectrograma Mel en escala logarítmica del fragmento, que se proporcionan como entrada a un codificador previo, que reducirá a la mitad la longitud del audio, preservando los detalles más importantes y disminuyendo considerablemente el costo temporal de ejecución del modelo. Posteriormente se obtienen también los *encodings* posicionales y se hace uso de bloques codificadores y decodificadores que estarán relacionados mediante mecanismos de atención. El modelo usará la palabra anterior que se ha predicho para mejorar la predicción de la siguiente. Se proporciona como salida el idioma al que pertenece el audio, la transcripción obtenida y alineamientos temporales de la transcripción, es decir, a qué instante de tiempo pertenece cada trozo de la transcripción. En la Figura 3.3 se puede encontrar una representación de la arquitectura de este modelo.

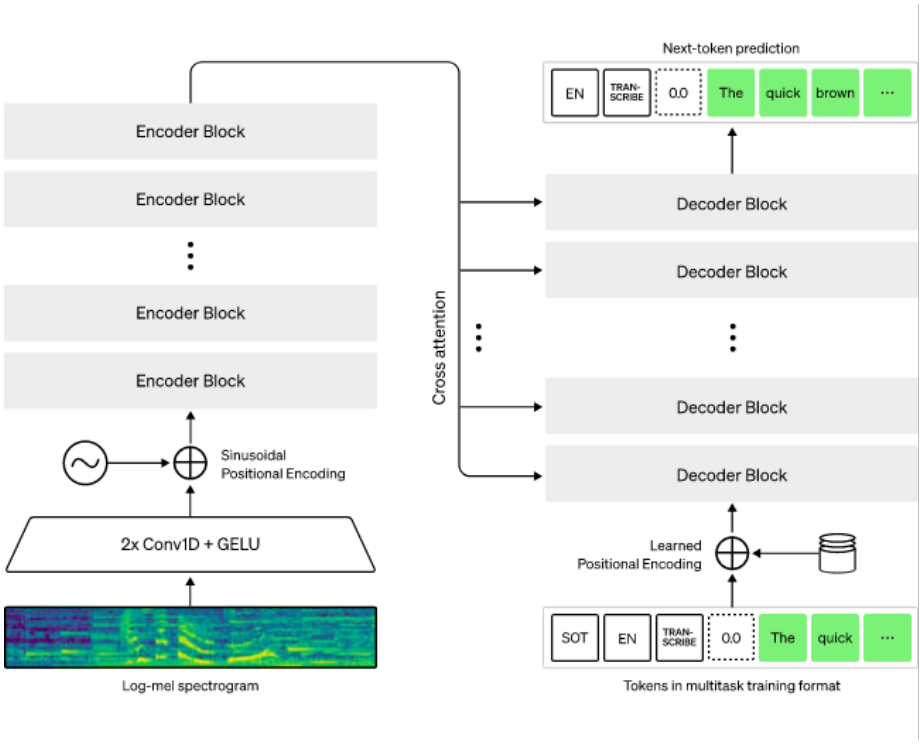


Figura 3.3: Arquitectura de Whisper [46].

Capítulo 4

Datos utilizados

4.1 LIP-RTVE

Con el objetivo de adaptar el modelo TalkNet-ASD, ya no sólo a la lengua española sino también a un escenario más cercano al caso de uso real de la herramienta final, hemos considerado adecuado hacer uso del conjunto de datos LIP-RTVE [47] para nuestro estudio.

Este conjunto de datos ha sido creado en base a distintos vídeos de telediarios de Radiotelevisión Española (RTVE), donde se extrajeron escenas en las que hubiera un hablante enfrente de la cámara. También cuenta con información adicional, como la posición de la cara del hablante y sus regiones de interés, las transcripciones pertenecientes a cada vídeo, así como el audio de cada vídeo en un fichero aparte.

Cuenta con aproximadamente 13 horas de vídeo en total, dividido en 323 hablantes diferentes, 163 de los cuales son hombres y 160 mujeres. Los vídeos presentan una resolución de 480 píxeles de ancho por 270 de alto a 25 fotogramas por segundo, una resolución considerablemente por debajo de la considerada alta definición, con lo que complica ostensiblemente la tarea de detectar el hablante en un vídeo dado, puesto que será más complejo analizar correctamente sus características faciales.

LIP-RTVE cuenta con dos tipos de particiones distintas de datos, ambas divididas en tres subconjuntos: entrenamiento, validación y test. En ambas particiones estos subconjuntos tienen aproximadamente 9 horas, 2 horas y 2 horas de vídeo, respectivamente. La primera partición es dependiente del hablante, donde los hablantes que aparezcan en una partición pueden aparecer en las demás. Este tipo de particiones resultan útiles para entrenar y evaluar modelos adaptados a ciertos individuos [48, 49, 50]. El segundo tipo de partición es independiente del hablante, donde cada uno de los subconjuntos cuenta con su propia lista de hablantes, que es mutuamente exclusiva con los otros subconjuntos. De esta forma, se pueden entrenar

modelos y evaluar fácilmente su robustez y capacidad de generalización al utilizar hablantes en validación y test distintos de los encontrados en entrenamiento.

4.2 Procesado de los datos

Hemos hecho uso de las mismas divisiones de los datos propuestas en la partición independiente del hablante, puesto que hemos querido entrenar un modelo que logre generalizar y, por lo tanto, pueda ser utilizado con cualquier otro hablante que no haya aparecido en el conjunto de entrenamiento.

Antes de poder hacer *fine-tuning* sobre el modelo TalkNet-ASD preentrenado, hemos tenido que procesar los datos a utilizar a un formato apto para proporcionárselos como entrada al modelo y rápido de cargar para acelerar el proceso de entrenamiento. Para ello, hemos realizado una serie de extracción de características y preprocesos.

Primeramente, se ha hecho uso del detector de caras DSFD [41] para detectar en cada fotograma de los vídeos la cara más grande, que en la gran mayoría de casos pertenece a la persona que está hablando. Para poder proporcionar estas imágenes como entrada al modelo, las *bounding boxes* de las caras extraídas fueron escaladas a 112×112 píxeles y convertidas a escala de grises. Tras obtener para cada fotograma del vídeo la cara más grande y procesarla, se guarda en un fichero comprimido (.npz), que permite reducir el espacio en disco ocupado por la información mientras que se mantiene una velocidad de carga elevada a la hora de entrenar el modelo.

Se ha procesado además el audio de cada vídeo, de manera que se ha procesado cada audio correspondiente para extraer los 13 Coeficientes Cepstrales en las Frecuencias de Mel (MFCC, por sus siglas en inglés) [51], tratando el audio a 100 hercios. De esta forma, dado que el vídeo viene representado a 25 fotogramas por segundo, podemos establecer una equivalencia de 4 elementos de audio por cada elemento de vídeo. Este alineamiento audiovisual es de vital importancia a la hora de abordar el entrenamiento del modelo para que éste pueda integrar ambas modalidades.

Estos dos pasos anteriores conllevan un costo temporal, puesto que, en total, este proceso de extracción de características ha tenido una duración de aproximadamente 25 horas para todos los vídeos del conjunto de datos. Cabe destacar que, posteriormente, se comprobó que las caras extraídas fueran las correctas, puesto que en algún caso aislado el hablante no era la persona con el rostro más grande del vídeo. Por el mismo motivo, se han procesado manualmente algunos de los vídeos para asegurar el buen rendimiento del modelo final. Sin embargo, no podemos asegurar que el conjunto de datos esté correctamente etiquetado en su totalidad, con lo que no podemos descartar que exista una pequeña cantidad de ruido a la hora

de entrenar y evaluar el modelo.

4.3 Creación de las muestras

Por último, se han creado las muestras a utilizar durante el entrenamiento del modelo, haciendo uso de la partición original independiente del hablante a la hora de crear los subconjuntos. Concretamente, hemos definido cuatro tipos de muestra, dividiendo las muestras negativas en tres subcategorías:

1. **Positivas:** Son aquellas locuciones en las que la cara se corresponde con la voz.
2. **Negativas:** Dentro de las muestras negativas contamos con tres subcategorías, en las que se modifica el audio de la muestra para que no haya una correspondencia entre voz y cara.
    - 2.1 **Desfase de audio:** Del mismo vídeo a analizar se toma un segmento de audio centrado en otro instante temporal, asegurando siempre que haya suficiente desfase temporal y no se solape con el audio correcto en más de cierto porcentaje. Enfocamos este tipo de muestra a que el modelo sea capaz de aprender a sincronizar mejor las características audiovisuales.
    - 2.2 ***Mismatch* parcial:** En este tipo de muestra tomamos un audio del mismo hablante, pero de otro de sus vídeos. Con este tipo de muestras el modelo debe aprender a entrelazar el audio y el vídeo para comprobar que realmente el audio se corresponde con los movimientos en la cara del hablante.
    - 2.3 ***Mismatch* completo:** En este caso, se hace uso de un audio de otro hablante (escogido aleatoriamente) del mismo subconjunto de datos. De esta manera, creemos que el modelo aprende a no memorizar o asociar el tipo de voz con un hablante en concreto, aportando así una mayor generalización.

Tomando en cuenta la partición de datos independiente del hablante de LIP-RTVE, se han generado muestras aleatorias para cada subconjunto. Concretamente, para cada muestra se ha elegido aleatoriamente su tipo entre los mencionados anteriormente, el índice del fotograma central a analizar en la muestra y, en caso de ser una muestra con *mismatch*, el audio a utilizar. Las muestras creadas están equilibradas entre clases, donde el 50 % son positivas y el otro 50 % negativas, siendo la probabilidad de que pertenezca a cualquiera de los tres grupos de muestras negativas equiprobable. Las muestras creadas se guardan en un archivo CSV separado para cada uno de los subconjuntos. El número de muestras en cada uno de ellos se muestra en la Tabla 4.1.

Subconjunto	Nº muestras creadas
Entrenamiento	100,000
Desarrollo	30,000
Test	30,000

Tabla 4.1: Número de muestras artificiales creadas por cada subconjunto.

4.4 Carga de los datos

Para poder entrenar el modelo y evaluarlo, surge la necesidad de tener un método rápido de cargar los datos y que, a su vez, realice todos los preprocesos necesarios para adaptar los datos de las muestras creadas a estructuras capaces de ser procesadas por el modelo.

El principal problema que se debe tener en cuenta en el momento de cargar los datos es la posibilidad de encontrarnos un vídeo o audio de una duración muy corta cuando deseamos entrenar un modelo que haga uso de un tamaño de contexto elevado. En esta situación no disponemos de suficientes datos como para rellenar el tamaño de entrada requerido por el modelo. Otra situación problemática es cuando el centro de la muestra es muy cercano al principio o al final del vídeo, donde también tenemos que ser capaces de adecuar los datos al modelo.

Por lo tanto, cuando se desea cargar una muestra, primero accede al archivo CSV del subconjunto correspondiente y se obtienen los nombres de archivo tanto del vídeo como del audio a usar, así como el índice del fotograma central de la muestra. Tras ello, se cargan los fotogramas de la cara, guardados previamente en formato npz, y se comprueba si, dado el tamaño de la ventana de contexto a utilizar, el centro de la muestra va a resultar en el uso de fotogramas previos al inicio del vídeo o posteriores a su final. En este caso, se tendrá que aplicar *padding* al lado donde falten datos, agregando matrices de ceros de 112×112, equivalente a las imágenes de caras; así, el modelo aprenderá a tratar con este relleno y a ignorarlo. Por último, se recorta al tamaño adecuado según el tamaño de contexto del modelo, de forma que el número de fotogramas a la izquierda y derecha del centro de la muestra sea equivalente.

Una vez cargado el vídeo y agregado el relleno necesario, se cargará el audio indicado por la muestra. En caso de ser una muestra positiva, simplemente alinearemos con el audio aplicando relleno por el mismo lado que en el vídeo (cabe destacar que un fotograma de vídeo equivale a cuatro conjuntos de características cepstrales por las tasas de muestreo a las que fueron tratados los datos, ver Sección 4.2). En caso de ser cualquier tipo de muestra negativa, deberemos seleccionar aleatoriamente un punto central sobre las características del audio. En caso de las muestras con *mismatch*, ya sea parcial o completo, este proceso se puede realizar con una selección

aleatoria de un índice. Sin embargo, con las muestras con desfase queremos asegurar que el audio y el vídeo no estén alineados. Para ello, debemos, por tanto, tener en cuenta que no haya más de cierto grado de solapamiento entre la posición del audio original (que realmente correspondería con el video) y el nuevo centro seleccionado. Hemos establecido el máximo de solapamiento en un 50 %, es decir, los datos extraídos con el nuevo centro no pueden compartir más de la mitad de las posiciones con el audio original.

Ya cargados y alineados los datos de vídeo y audio a utilizar, se deben crear las etiquetas de clase usadas por el modelo para conocer si una muestra es positiva o negativa a la hora de clasificarla y aprender posteriormente. Dado que el conjunto de datos no nos proporciona información fotograma a fotograma de pausas en el habla o momentos en los que no hay habla, asignaremos a cada fotograma una etiqueta, siendo la misma que venga indicada por la muestra artificial, positiva o negativa, puesto que TalkNet-ASD fue diseñado para proporcionar como salida una predicción de clase por cada fotograma de entrada. Puesto que la base de datos LIP-RTVE está enfocada al reconocimiento del habla, únicamente hay una persona hablando en cada vídeo, por lo que podemos asignar la misma etiqueta a todos los fotogramas que componen la muestra.

Terminado todo este proceso, los datos se pueden proporcionar como entrada directamente al modelo para su estimación.

Capítulo 5

Experimentación

Una parte fundamental de este proyecto ha sido la experimentación sobre el modelo de detección del habla, donde ponemos a prueba las hipótesis planteadas y se analizan los resultados obtenidos por el modelo. El objetivo de este capítulo es proporcionar una evaluación exhaustiva del modelo para poder entender cómo lograr el mayor rendimiento posible con los datos de los que disponemos. El núcleo central de nuestro estudio se basa en realizar un *fine-tuning* del modelo ya pre-entrenado con el conjunto de datos AVA-ActiveSpeaker, puesto que nosotros disponemos de un conjunto de datos relativamente pequeño y creemos que podemos obtener un mejor rendimiento de esta manera que entrenando desde cero.

5.1 Factor de aprendizaje y número de *epochs*

Primeramente, hemos probado diferentes valores de factores de aprendizaje, aprovechando al mismo tiempo para ajustar el número de *epochs* de entrenamiento, analizando cuánto tarda el modelo en aprender las características de los datos y lograr un buen rendimiento. En la Figura 5.1, se muestran las diferentes tasas de aprendizaje utilizadas y la evolución de la métrica mAP (descrita en la Sección 2.3) a nivel de fotograma en el conjunto de validación a lo largo de los *epochs* que conforman el entrenamiento. Cabe destacar que esta métrica se ha analizado a nivel de fotogramas y no de muestras, es decir, contabilizando las clasificaciones de cada fotograma de forma individual. Para estos experimentos hemos utilizado el optimizador Adam junto con el *scheduler* StepLR, junto con una ventana de contexto de 25 fotogramas de tamaño. También se probó a utilizar un *scheduler* coseno, pero se obtenían resultados peores.

Se puede observar en esta figura que la tasa de aprendizaje que mejores resultados de mAP logra en el conjunto de datos de validación es 0.0001, mostrando una ventaja considerable con respecto al resto de valores con los

que se ha experimentado. Igualmente, se puede ver que a partir del séptimo u octavo *epoch* ya no se consiguen mejores resultados, excepto con el factor de aprendizaje más bajo de todos, puesto que se ve forzado a aprender sobre los datos a un ritmo muy lento. Por tanto, para los siguientes experimentos llevados a cabo hemos hecho uso de 9 *epochs*, permitiéndonos tener cierto margen donde aún podrían mejorar los resultados, pero al mismo tiempo evitando un posible sobre-entrenamiento al aplicar un mayor número de *epochs*.

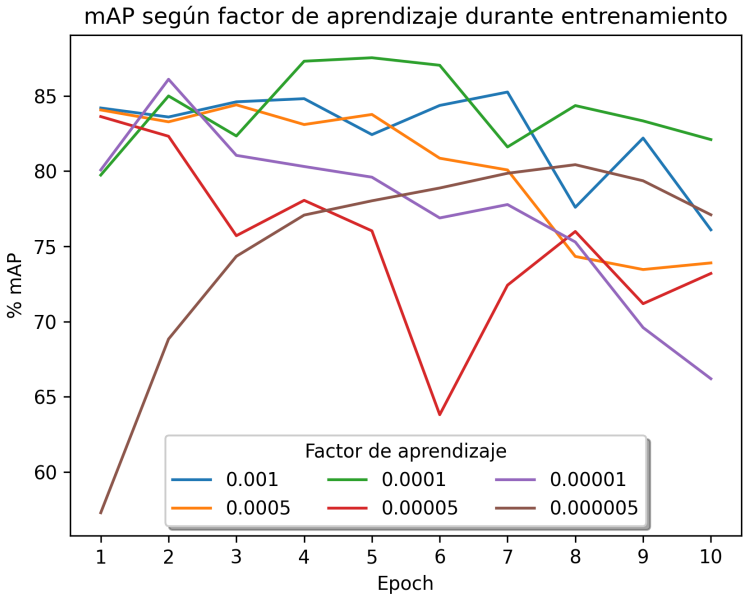


Figura 5.1: Evolución de la métrica mAP en el conjunto de validación a lo largo de los *epochs* en función del factor de aprendizaje.

Se ha tenido en cuenta para los experimentos de esta sección el intervalo de confianza del 95 % a la hora de mostrar los resultados de tasa de acierto y evaluar los resultados, puesto que podrían haber solapes entre las tasas de acierto obtenidas por distintos valores del factor de aprendizaje, potencialmente modificando la elección del mejor factor de aprendizaje a usar.

En la Tabla 5.1 se encuentran las métricas de rendimiento analizadas para los distintos valores de factor de aprendizaje con los que se han experimentado.

Factor de aprendizaje	mAP	Tasa de acierto
0.001	84.8	78.7±0.5
0.0005	84.4	90.7±0.3
0.0001	**87.5**	**91.8±0.3**
0.00005	83.6	88.1±0.4
0.00001	86.1	87.1±0.4
0.000005	80.9	83.4±0.4

Tabla 5.1: Mejores valores de tasa de acierto y mAP obtenidos para cada factor de aprendizaje en el conjunto de validación.

Tras analizar los resultados obtenidos, destaca claramente el rendimiento resultante de entrenar con una tasa de aprendizaje de 0.0001, que logra unos resultados considerablemente mejores que los otros valores probados. Además la tasa de acierto obtenida teniendo en cuenta el intervalo de confianza no se solapa con ningún otro resultado, con lo que podemos tener la certeza de que es el mejor valor del que podemos hacer uso. Por este motivo, hemos utilizado este valor en nuestros siguientes experimentos de ajuste de parámetros.

5.2 Ventana de contexto

Otro factor muy importante que afecta al rendimiento de los modelos entrenados es el tamaño de la ventana de contexto a la hora de conformar las muestras utilizadas durante el entrenamiento. Una primera hipótesis, de acuerdo con los resultados obtenidos en el artículo original [37], sería que cuanto más contexto y datos posea el modelo sobre la muestra de entrada, más características podrá utilizar y se podrá tomar una decisión de clasificación más informada. Por este motivo, decidimos explorar con múltiples tamaños de ventana al entrenar. Los resultados obtenidos en este estudio quedan reflejados en la Tabla 5.2.

Observando detenidamente la Tabla 5.2, se ve una tendencia ascendente del rendimiento del modelo a medida que se aumenta el tamaño de la ventana de contexto, respaldando los resultados de otros estudios [37]. Los peores resultados se obtuvieron con una ventana pequeña de 5 fotogramas y los mejores con una de 51, la cual representa dos segundos de duración de vídeo a 25 fotogramas por segundo. El contexto con el que cuenta el modelo a la hora de clasificar es muy importante a la hora de detectar el hablante, ya que si el contexto es muy pequeño, el modelo no logra analizar adecuadamente los diferentes patrones del habla o características visuales que puedan haber. Sin embargo, al definir ventanas de mayor tamaño, el modelo cuenta con una elevada cantidad de contexto para clasificar de forma informada la muestra. Aunque la diferencia entre una ventana de contexto de 43 y 51

fotogramas de tamaño es mínima.

Tamaño de ventana	mAP	Tasa de acierto	AUC
5	60.0	70.2±0.5	0.76
9	72.0	79.2±0.5	0.84
13	77.7	83.5±0.4	0.89
17	78.4	85.7±0.4	0.92
21	85.2	89.9±0.3	0.94
25	87.5	91.8±0.3	0.96
35	90.7	93.1±0.3	0.98
43	94.8	97.0±0.2	**0.99**
51	**95.8**	**97.3±0.2**	**0.99**

Tabla 5.2: Mejores valores de mAP, tasa de acierto y AUC obtenidos para cada tamaño de ventana en el conjunto de validación.

5.3 Tiempo de inferencia

Un factor importante a considerar es el tiempo de inferencia del modelo al tomar como entrada muestras o segmentos de diferentes tamaños, puesto que si representan una diferencia significativa, este aspecto debe tenerse en cuenta a la hora de elegir el modelo a utilizar en la herramienta. Hemos analizado los tiempos de inferencia según el tamaño de ventana de entrada medido en fotogramas. Para ello, hemos clasificado una misma muestra 10.000 veces modificando el tamaño, eliminando así cualquier cuello de botella que pueda existir por parte del procesador al tener que cargar las muestras y procesarlas, puesto que este proceso se realiza solo una vez. Se ha repetido el experimento 5 veces y se ha calculado la media para cada tamaño de entrada. En la Tabla 5.3, se encuentra la media en segundos de las ejecuciones para cada tamaño de ventana de las muestras de entrada.

Nº fotogramas muestra	Tiempo de inferencia (s)
5	395
13	406
25	405
51	397

Tabla 5.3: Comparación de tiempos de inferencia entre distintos tamaños de entrada.

Los resultados indican que la diferencia entre analizar segmentos de vídeo de distintas longitudes es insignificante o nula, puesto que las diferencias mostradas están razonablemente dentro del margen de error. Por lo tanto,

se concluye que el uso de cualquier tamaño de ventana a la hora de analizar un vídeo con la herramienta nos es válida y el factor más importante es la calidad de análisis y clasificación del modelo a utilizar.

Probamos también a realizar el experimento analizando el conjunto de validación al completo, pero sí que habían diferencias debido a la carga y procesado de los datos, que es más costoso al tener una mayor longitud. Sin embargo, esto no es un problema que se vaya a dar en la herramienta final, puesto que se procesará todo el vídeo al principio, tal y como se describe en el Capítulo 6 de la memoria.

5.4 Comparación entre los mejores modelos

Posteriormente, hemos probado los dos modelos que consideramos más interesantes en el conjunto de test para poder evaluar su rendimiento con un conjunto de datos diferente y concluir la selección de modelos. En la Tabla 5.4 se encuentran los resultados obtenidos.

Nº *frames* de muestras del modelo	**mAP**	**Tasa de acierto**	**AUC**
13	68.3	81.4±0.4	0.89
25	84.2	89.3±0.4	0.91
35	86.8	90.1±0.4	0.98
51	91.6	95.4±0.2	0.99

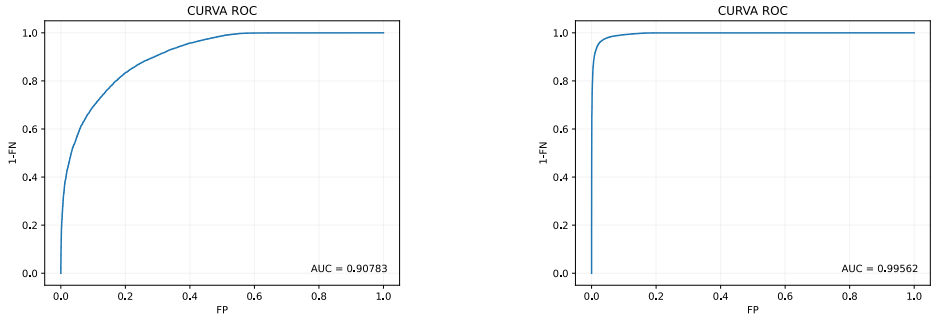
Tabla 5.4: Tasa de acierto, mAP y AUC obtenidos en el conjunto de datos de test.

Ambos modelos funcionan muy bien, pero ya que buscamos que las muestras sean estables y todo lo precisas posibles, creemos que el modelo de 51 fotogramas de contexto puede funcionar mejor en nuestro caso de uso para la herramienta final.

5.5 Umbral de clasificación óptimo

Es importante ajustar este modelo para que clasifique correctamente. Por otro lado, nos interesa que el modelo tenga cierta consistencia a la hora de clasificar, puesto que una predicción negativa en medio de una muestra positiva puede significar acortar considerablemente la muestra final y disponer de menos información en la base de datos creada. Para arreglar este problema, vamos a estudiar los diferentes umbrales de clasificación que podríamos usar para mejorar la estabilidad de la clasificación y asegurar cierta continuidad en las muestras creadas.

Con el objetivo de analizar los umbrales, hemos clasificado el conjunto de test y obtenido las probabilidades de que cada fotograma analizado sea positivo [52]. Tras ello se ha construido una curva ROC para los dos modelos candidatos, es decir, aquellos entrenados con ventanas de 25 y 51 fotogramas de contexto. Mediante estas curvas ROC podemos ir comprobando cada uno de los posibles umbrales de clasificación representados por la curva para intentar obtener un balance entre el mínimo número de falsos positivos y de falsos negativos, mejorando así el rendimiento del modelo, puesto que el umbral por defecto de 0.5 que utiliza el modelo TalkNet-ASD no es necesariamente el mejor para nuestro caso de uso. En la Figura 5.2, se encuentran las curvas ROC generadas por ambos modelos al clasificar el conjunto de test.



(a) 25 fotogramas (b) 51 fotogramas

Figura 5.2: Curvas ROC obtenidas en test para modelos con ventanas de contexto compuestas de 25 y 51 fotogramas.

Se ha analizado cada posible umbral para determinar la mejor puntuación f1 posible y se ha obtenido que para el modelo con menor contexto el mejor umbral es 0.19 y para el modelo con mayor contexto 0.03. Lógicamente, reducir tanto el valor del umbral de decisión puede suponer un aumento considerable en los falsos positivos a la hora de clasificar, pero también disminuirá en gran medida los falsos negativos, lo que nos interesa puesto que si no se dispone de muchos vídeos de entrada para crear un conjunto de datos es importante poder utilizar toda la información disponible.

5.6 Pre-entrenamiento vs. *Scratch*

Hemos probado también a entrenar el modelo desde cero con una inicialización de los pesos aleatoria en entrenamiento, es decir, sin hacer uso del modelo pre-entrenado que proporcionan los autores de talkNet-ASD con el conjunto de datos AVA-ActiveSpeaker [18]. De esta forma, podemos estudiar hasta qué punto la inicialización del modelo puede llegar a afectar el

rendimiento de éste sobre nuestro conjunto de datos. En la Figura 5.3 podemos encontrar la gráfica comparativa entre las dos formas de inicializar los pesos del modelo. En el caso del modelo con pesos iniciales aleatorios, cabe destacar que hemos usado una tasa de aprendizaje de 0.0005, puesto que entrenar desde cero requiere normalmente de una tasa de aprendizaje mayor.

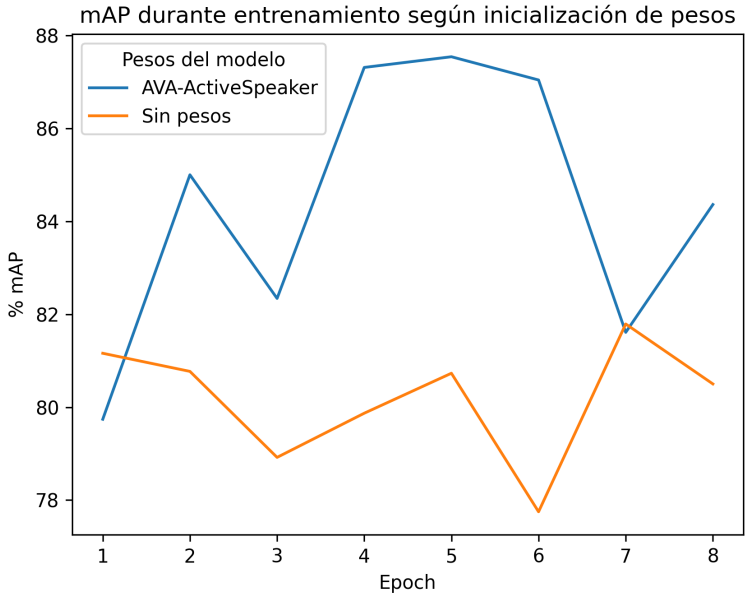


Figura 5.3: Evolución de la métrica mAP a lo largo de los *epochs* según los pesos iniciales.

Examinando los resultados, queda clara la ventaja de hacer uso de los pesos ya propuestos por los autores del modelo, puesto que el modelo ya ha aprendido a analizar las características audiovisuales con un corpus de gran tamaño. De este modo, adaptarlo a nuestra tarea resulta mucho más sencillo que entrenarlo con una inicialización de pesos aleatoria.

Además de esto, hemos comparado los resultados obtenidos por los modelos a los que le hemos realizado un *fine-tuning* con el corpus LIP-RTVE para cada tamaño de ventana de contexto y el modelo pre-entrenado con el conjunto de datos AVA-ActiveSpeaker por los autores de TalkNet-ASD. Con esta comparación queremos comprobar qué tan bien se ha adaptado el modelo a nuestros datos y si es necesario todo el proceso de experimentación y entrenamiento que se ha mostrado en este capítulo. Para ello hemos clasificado el conjunto de validación de LIP-RTVE utilizando distintos tamaños de ventana, cada uno con su modelo correspondiente en el caso de los re-entrenados con LIP-RTVE. De este proceso, hemos obtenido la tasa de acierto a nivel de fotograma, mostrada en la Figura 5.4.

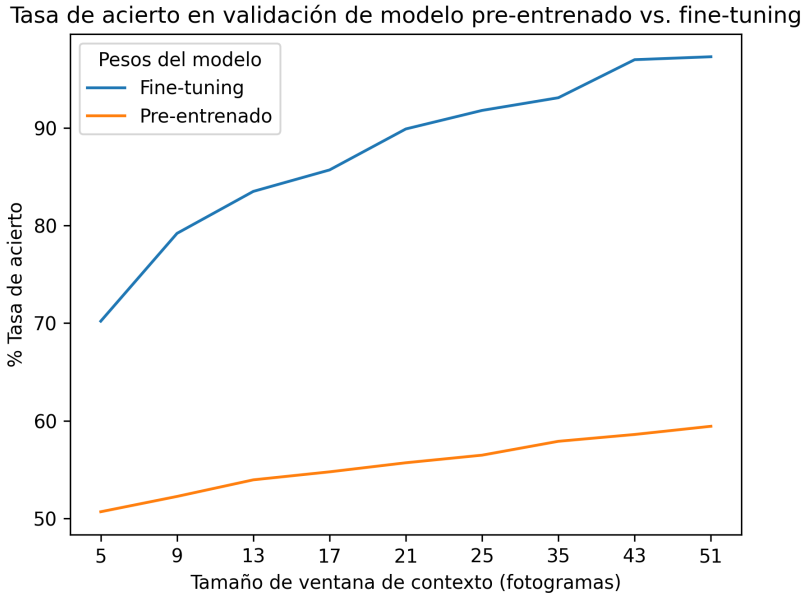


Figura 5.4: Comparación de tasa de acierto entre modelo pre-entrenado y *fine-tuned* con el corpus LIP-RTVE en validación.

Se observa una inmensa diferencia entre el modelo pre-entrenado y cada uno de los modelos re-entrenados para un tamaño de ventana específico con LIP-RTVE. Por tanto, podemos afirmar que para obtener buenos resultados es imperativo el proceso de adaptación del modelo a nuestro conjunto de datos.

Capítulo 6

Estructura de la herramienta

En este capítulo, se explicará en detalle el funcionamiento y arquitectura de la herramienta, así como el proceso por el que pasan los datos de entrada suministrados. La herramienta desarrollada a lo largo de este trabajo cuenta con dos partes: un primer *script* de Python encargado de tomar como entrada un vídeo o directorio de vídeos, analizarlos por completo y producir datos de muestras que serán interpretados por la segunda parte de la herramienta, la interfaz final, utilizada por el usuario para comprobar las muestras creadas y verificar su calidad. Esta interfaz permitirá tanto aceptar como rechazar la muestra, o incluso modificar la transcripción generada automáticamente, tal y como puede observarse en la Figura 6.3.

6.1 *Pipeline* de detección del habla

La primera parte de la herramienta consta de un *script* de Python, que se encarga de todo lo relacionado con tomar los vídeos o el directorio donde se encuentran, procesarlos, y crear las muestras correspondientes que serán posteriormente comprobadas por el usuario anotador de datos mediante la interfaz creada para la herramienta. En la Figura 6.1 se encuentra un diagrama que ilustra los distintos procesos por los que pasan los datos y la arquitectura de este *script*.

Cabe destacar que esta herramienta cuenta con una serie de opciones disponibles al ejecutarla desde la consola:

- *–minLength*: Longitud mínima en fotogramas para considerar que un segmento de hablante representa una muestra.

- *–clThreshold*: Umbral que determinará si una muestra se clasificará como positiva o negativa a la salida del modelo, 0.5 por defecto.

- *–outputVideo*: Si se introduce esta opción, el *script* proporcionará como salida adicional el vídeo completo etiquetado.

- *–model*: Nombre del fichero del modelo a utilizar, el cual, a no ser que se modifique el código fuente, debe cumplir con el formato de entrada y salida de talkNet-ASD.

- *–windowSize*: Tamaño de ventana (en fotogramas) a utilizar por el modelo a la hora de clasificar las muestras.

- *–whisperLang*: Idioma en el que se ejecutará el modelo de reconocimiento del habla Whisper; por defecto, el modelo detectará automáticamente el lenguaje.

- *–minMethod*: Cambia el método de detección del modelo a una clasificación fotograma a fotograma.

- *–smoothWindowSize*: Tamaño de la ventana (en fotogramas) de postprocesado para suavizar posibles inconsistencias de clasificación que puedan existir.

Estas opciones de ejecución permiten una experiencia de usuario adaptable y personalizada, ya que en cualquier momento se puede trabajar con un modelo distinto, modificar el idioma de la transcripción o ajustar parámetros que permiten modificar las muestras obtenidas.

Como única entrada obligatoria, este programa requerirá el directorio donde se encuentran los vídeos o el nombre completo del vídeo a analizar. No se requiere ningún tipo de distinción por parte del usuario entre los dos tipos de entrada, puesto que el *script* detectará automáticamente si se trata de un directorio o de un vídeo a la hora de procesarlo.

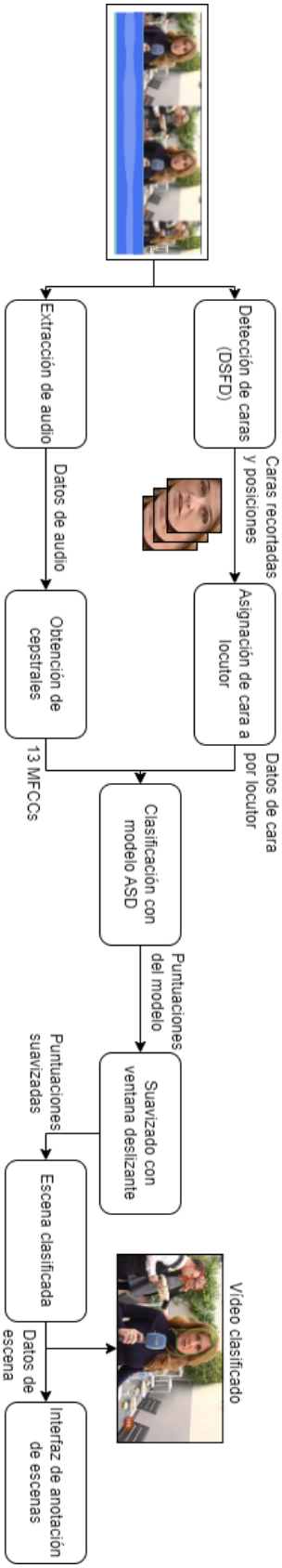


Figura 6.1: Esquema de la arquitectura interna de la herramienta.

6.1.1 Procesado del vídeo

Tras ello, para cada vídeo se aplicará el proceso completo de transformación de los datos y clasificación de los mismos. En primer lugar, tal y como refleja la Figura 6.1, se recorre cada fotograma del vídeo y se aplica el modelo detector de caras DSFD, el cual obtendrá las localizaciones y tamaños (*bounding boxes*) para cada cara existente en ese fotograma. Una vez obtenidos los datos de cada cara se sigue el siguiente proceso:

Al analizar el inicio del vídeo, se guarda la localización y dimensiones de cada cara, así como el número de fotograma en el que aparece. Al analizar los siguientes fotogramas, es de vital importancia asignar la misma cara a cada persona, puesto que inconsistencias en la detección de cada individuo pueden resultar en errores en la muestra, como falsos positivos o falsos negativos, o incluso una muestra con caras que no se corresponden únicamente a la misma persona. La solución propuesta para ello ha sido hacer uso de la distancia euclídea [53] entre las *bounding boxes* del fotograma actual y el fotograma anterior. De esta manera, si una cara detectada consta de una posición y tamaño muy similar a otra del fotograma anterior, será asignada a la misma persona. Sin embargo, en caso de alejarse más de cierta distancia umbral en píxeles de todas las caras del fotograma anterior, será considerada una persona nueva en el fotograma analizado, ya sea porque esa persona ha entrado en escena o porque ha habido un cambio de plano. Para calcular este umbral, tomamos en cuenta las dimensiones del vídeo, es decir, la cantidad de píxeles de altura y de anchura. Posteriormente, tal y como se detalla en la ecuación 6.1, dividimos el producto de ambas dimensiones para obtener un valor normalizado. De esta forma, se tienen en cuenta las diferencias de resolución entre vídeos, puesto que no es lo mismo una distancia de 50 píxeles en un vídeo de baja resolución que en uno de resolución 4K.

Tras probar exhaustivamente con diferentes resoluciones de vídeo, hemos concluido que el mejor valor percibido de normalización es 2500, permitiendo cierto margen para movimientos rápidos de una persona en el plano pero con suficiente potencia discriminativa como para tener en cuenta nuevas personas en el fotograma. La fórmula utilizada se puede observar en la ecuación 6.1.

$$Umbral_{dist} = \frac{(Ancho_v \times Alto_v)}{2500} \tag{6.1}$$

Adicionalmente, en caso de que una cara no aparezca y luego vuelva a aparecer ciertos fotogramas más tarde será considerada como una nueva persona; esto se debe a que nuestro interés es crear bases de datos para tareas como lectura de labios automática [54, 55], necesitando una visualización continua sobre la cara de la persona. Así mismo, la ausencia de una cara y su aparición algunos fotogramas más tarde puede simbolizar un cambio de

plano, que no queremos que sea asociado a una persona incorrecta.

Hemos probado también a realizar la agrupación de individuos mediante la distancia euclídea de la propia cara, con el objetivo de clasificar según los posibles rasgos faciales de las distintas personas, pero debido a la baja resolución de los vídeos del conjunto de datos en algunas ocasiones puede producir clasificaciones incorrectas, añadiendo ruido no deseado a las muestras generadas por la herramienta. Además, la comparación de las distancias entre las posiciones y tamaños de las caras resulta más eficiente de calcular, puesto que se trata de únicamente 4 coordenadas por cara.

A medida que se vayan agrupando las caras, no solo se obtendrá la información de las localizaciones y tamaños de las mismas, sino que se recortará en cada fotograma el espacio correspondiente a ese rostro. Posteriormente, se realiza un escalado a 112×112 y se convertirá a escala de grises, asegurando la compatibilidad con el modelo (TalkNet-ASD, en nuestro caso). De igual manera, obtendremos datos de los fotogramas en los que aparece cada cara, que serán de gran utilidad posteriormente para alinear el audio con el vídeo a la hora de analizar cada potencial hablante por separado.

Obtenida ya toda la información pertinente a cada cara y hablante, el siguiente paso será extraer el audio del vídeo. Para ello usaremos la herramienta de software libre FFmpeg [56]. Este audio extraído será procesado y adaptado al formato de entrada del modelo, obteniendo 13 coeficientes cepstrales de Mel del mismo [57].

6.1.2 Clasificación con el modelo

Habiendo recabado los datos necesarios para proporcionar como entrada al modelo de detección del habla, se cargará el modelo talkNet-ASD que hemos entrenado para una ventana de 51 fotogramas. En caso de haberse especificado una ruta de modelo por parte del usuario, cargaremos el modelo indicado del fichero seleccionado, el cual debe contener datos tanto de la arquitectura del modelo como de sus pesos para poder funcionar adecuadamente. Se seleccionará también el tamaño de ventana a aplicar, por defecto 51 fotogramas, parámetro también modificable por el usuario al ejecutar el programa.

Cargado ya el modelo, es hora de clasificar los datos extraídos del vídeo y audio, para lo cual clasificaremos cada persona por separado. Procesamos el audio y el vídeo, alineándolos según el momento en el que aparece el potencial locutor. Hemos probado con varias formas de realizar esta clasificación sobre el vídeo:

- **Método secuencial:** Es el caso más intuitivo y fácil de implementar. Se trata de tomar la ventana de clasificación e ir moviéndola en un *stride* de tanta longitud como longitud tenga la propia ventana. De esta forma vamos clasificando secuencialmente los "trozos" de vídeo

procesado sin ningún tipo de solape ni contexto del anterior. En base a nuestras pruebas, este método presenta una buena calidad de clasificación y consistencia, aunque cuenta con el problema de clasificar normalmente todo el segmento con la misma etiqueta, debido a la naturaleza de nuestros datos de entrenamiento. Por tanto, si una persona habla durante un segundo pero usamos una ventana de clasificación con una longitud de dos segundos, tiende a clasificar todos esos dos segundos con la misma etiqueta, cuando esto no sería correcto.

- **Método medias:** Para combatir el problema de clasificar un segmento de vídeo con una única etiqueta, hemos ideado otro método de clasificación. Al igual que el anterior método, en primer lugar, procesamos la ventana de contexto entera con el modelo. No obstante, una vez tengamos una etiqueta por cada fotograma, calcularíamos la media entre ellos, asignando el resultado únicamente al fotograma sobre el que se centra la ventana de contexto actualmente. Posteriormente, se avanza un fotograma y se repite el proceso, obteniendo la media de la clasificación nuevamente. De esta forma se va clasificando cada fotograma poco a poco y se aprovecha el contexto de los fotogramas adyacentes y su información mediante la media. Esta aproximación presenta dos problemas. El primero son los posibles "saltos" de clasificación que pueden darse, donde un fotograma puede ser clasificado como positivo y el siguiente como negativo aunque la persona estuviese haciendo lo mismo que en el fotograma anterior. El segundo problema es la complejidad temporal que presenta este método en comparación con el secuencial, ya que este último avanza en cada clasificación tanto como longitud tenga la ventana. Sin embargo, ahora clasificamos fotograma a fotograma, lo que ralentiza considerablemente todo el proceso.

- **Método mínimo:** Otra posible aproximación que hemos probado es la clasificación "mínima", que resulta muy similar en funcionamiento al método de medias, puesto que vamos clasificando únicamente el elemento central de la ventana y se va moviendo la ventana fotograma a fotograma. Sin embargo, esta vez no tenemos en cuenta el contexto de los fotogramas adyacentes de forma explícita, ya que no hacemos cálculos de medias ni post-procesado, puesto que únicamente se toma el resultado de ese fotograma central devuelto por el modelo sin ningún retoque adicional. En este caso, esperamos que el modelo sea capaz de integrar el contexto de los fotogramas adyacentes y devolver un resultado correcto. Cuenta con los mismos problemas que el método de medias, puesto que es lento y puede producir algunos resultados inconsistentes, pero funciona mejor y subsana parcialmente el problema de clasificar toda la ventana con una etiqueta, ya que internamente el modelo seguirá teniendo en cuenta el contexto de los

fotogramas adyacentes para clasificar con una única etiqueta, pero al mover de uno en uno el fotograma a clasificar reduce la problemática que surge.

En la práctica, el método secuencial es, en base a nuestras pruebas, el que mejor funciona, siendo, además, el más sencillo de implementar y rápido. Aunque usualmente clasifique con una etiqueta todo el tramo, no es algo que ocurra siempre, y se alivia enormemente el problema al usar tamaños de ventana menores que aportan una mayor flexibilidad ante cambios de patrones. Por tanto, es este método el que se usa por defecto en este programa. Sin embargo, cabe destacar que el método mínimo también proporcionaba unos resultados aceptables, por lo que el usuario puede seleccionarlo al ejecutar la herramienta mediante la opción "–*minMethod*".

6.1.3 Suavizado de probabilidades

Un problema que surge con todas las aproximaciones son las pequeñas inconsistencias de uno o dos fotogramas en los que el modelo clasifica una escena positiva como negativa y viceversa, descartando la muestra y cortándola por la mitad. Para aliviar este problema hemos creado una función de post-procesado sobre la clasificación. En esta función, hacemos uso de una ventana deslizante de corta longitud para "suavizar" las probabilidades de que una escena sea positiva tras la clasificación. Esta ventana comenzará en el primer fotograma clasificado del hablante y le asignará el valor de la media de todas las probabilidades de etiqueta positiva de los fotogramas que se encuentren dentro de la ventana. Este proceso se repite avanzando de un fotograma en fotograma. Esto elimina, en gran parte, estas inconsistencias o "saltos" que se puedan producir y mejora la calidad de las muestras generadas. Tras cierta experimentación, hemos encontrado que el mejor valor de la ventana de suavizado bajo nuestro criterio, es de 11 fotogramas. No obstante, es personalizable por el usuario con la opción "–*smoothWindowSize*".

6.1.4 Salida del *pipeline*

Habiendo obtenido ya todos los datos de la puntuación suavizada para cada persona en cada fotograma en el que aparece, es momento de la clasificación final. Para ello, se recorre cada puntuación obtenida y se binariza mediante un umbral de decisión. Si la puntuación supera este umbral, la muestra será positiva, mientras que en caso contrario será negativa. Esto nos ayuda a regular la clasificación, puesto que si tenemos un umbral con un valor menor, probablemente obtengamos segmentos más largos y menos falsos negativos. Sin embargo es importante tener en cuenta que bajar el valor del umbral tenderá a crear más falsos positivos. Cabe destacar que, al tratarse de una aplicación de anotación de datos, es preferible poder aprovechar,

en la medida de lo posible, el vídeo o conjunto de datos a analizar. Por lo tanto, preferiremos tener más falsos positivos que falsos negativos. El valor de este umbral es ajustable mediante la opción "*–clThreshold*".

Finalizado el proceso de clasificación, se pasa al montaje de los datos de salida que proporcionará el *script*. En caso de que el usuario desee obtener como salida una visualización de la clasificación de sus vídeos, puede indicarlo con el parámetro "*–outputVideo*". De ser así, se toman todos los resultados de clasificación, así como sus correspondientes puntuaciones, para dibujar las *bounding boxes* en las caras correspondientes, variando con un gradiente de colores entre rojo y verde según qué tan seguro esté el programa de que esa cara corresponda a una muestra negativa o positiva. Además, encima de cada cara aparece la probabilidad de que esa persona esté hablando. Se muestra un ejemplo de un fotograma de vídeo de salida en la Figura 6.2.

Figura 6.2: Ejemplo de un fotograma extraído de un vídeo tras clasificarlo con la herramienta.

La salida principal del vídeo serán las escenas etiquetadas para su revisión en la interfaz de la herramienta. Primero, inicializaremos el modelo de reconocimiento de habla Whisper y lo aplicaremos en el audio, extrayendo, para cada escena del vídeo, el segmento acústico adecuado según los instantes de inicio y finalización de la escena detectada. Una vez obtenidas las transcripciones de cada muestra, guardaremos los datos obtenidos del vídeo en un fichero separado. Estos datos incluyen, entre otros, los rostros recortados, la transcripción o en qué fotograma aparece cada persona. Tomaremos los datos procesados y los guardaremos en un fichero de salida en formato CSV, donde contaremos con las siguientes columnas:

- Video: Nombre del vídeo del que proviene la muestra.
- Speaker: ID dependiente de la escena del hablante que aparece en la muestra, relacionado con los datos guardados previamente.

- Ini: Marca temporal en la que se inicia la muestra dentro del vídeo (en segundos).
- End: Marca temporal en la que finaliza la muestra dentro del vídeo (en segundos).
- DataPath: Directorio del fichero que alberga los datos respectivos al vídeo.
- Transcription: Transcripción de la muestra generada por Whisper.

Este fichero será interpretado por la interfaz para crear muestras visuales y enseñárselas al usuario anotador, logrando una mejor comprensión sobre los datos.

6.2 Interfaz de usuario

La indispensable segunda parte de la herramienta es la interfaz de usuario, ejecutable mediante el fichero "gui.py". Haciendo uso de ella, se podrán analizar las muestras creadas por el *script* anteriormente descrito. Ha sido creada mediante la librería de Python "CustomTkinter"[1], que ofrece amplias posibilidades de personalización e interacción con la interfaz. En la Figura 6.3, se puede observar la apariencia que toma la interfaz gráfica de nuestra herramienta.

Figura 6.3: Interfaz de la aplicación desarrollada.

[1] `https://github.com/TomSchimansky/CustomTkinter`

Como se puede ver, consta de una ventana para poder reproducir la escena detectada, un campo de texto que posibilita la visualización y edición de la transcripción automática y distintos botones para habilitar la interacción con la herramienta. Además, se han introducido unas breves instrucciones de las teclas rápidas disponibles para poder interactuar con la reproducción del vídeo.

Al iniciar la aplicación, ésta abrirá el fichero CSV de muestras generadas anteriormente por la primera parte de la herramienta. Entonces se analizará la primera fila del fichero, obteniendo datos interpretables por la interfaz para generar una visualización de la escena. Tras leer estos datos se carga el vídeo original al que pertenece, se recorta el segmento de vídeo al que pertenece la escena y se dibuja un cuadrado verde alrededor del rostro de la persona locutora para poder identificarla. Todo esto se mostrará en el panel de vídeo de la aplicación gracias a la librería "python-vlc"[2], que proporciona una API para la interacción con el conocido reproductor de vídeo VLC media player [58].

Se ha preferido realizar la generación de estas visualizaciones de manera dinámica en vez de guardarlas en disco al clasificar los vídeos con el *script* inicial, puesto que podrían ocupar una cantidad de memoria substancial, aumentando en gran medida dependiendo de la duración y resolución del vídeo de entrada. Sin embargo, generar estos vídeos sobre la marcha supone un proceso casi instantáneo y nada costoso, donde el usuario no notará ningún tiempo de espera que retrase su trabajo en la anotación de los datos.

Justo debajo de esta visualización generada se integra la transcripción obtenida, representada mediante el cuadro de texto en la parte inferior de la aplicación para que pueda ser observado y corregido por el usuario, ya que todo el texto que aparece en este cuadro es editable.

En la parte derecha de la interfaz podemos encontrar dos botones. El botón verde "Accept", nos permite aceptar la escena, cuyos datos serán copiados en un fichero CSV aparte como muestra definitiva del conjunto de datos a construir. Tomará en cuenta adicionalmente cualquier cambio que haya realizado el usuario anotador sobre la transcripción generada por Whisper, guardando los cambios realizados en la caja de texto. Al aceptar una escena, se pasará automáticamente a la siguiente. En caso de hacer click sobre el botón rojo "Incorrect", se descartará la escena actual y se continuará al análisis de la siguiente escena del fichero.

En la parte inferior de la interfaz se hallan dos botones de navegación por las escenas. El botón "Prev" permitirá volver a la muestra anterior. Esto puede resultar de interés si se ha clasificado mal y se desea realizar un cambio, de manera que la herramienta permite subsanar errores humanos. En caso de haber aceptado manualmente una muestra como correcta, si tras volver a ella se decide que no alcanza la calidad esperada o que la

[2] `https://wiki.videolan.org/Python_bindings/`

persona indicada en el vídeo no está hablando, se puede seleccionar como incorrecta, eliminándose por tanto del fichero CSV de salida. A su derecha se encuentra el botón "Next", que permite pasar a la siguiente muestra sin clasificar la actual; en caso de haberla clasificado previamente, simplemente se mantendrá la clasificación indicada por el usuario. Este botón es útil cuando no se tiene claro si una muestra es correcta o no y se quiere seguir sin perder tiempo, o si se han retrocedido varias muestras con el botón "Prev" y se desea continuar el proceso sin tener que reclasificar muestras anteriores.

Adicionalmente, se podrán controlar ciertas funciones del reproductor de vídeo mediante atajos de teclado, indicados en la sección inferior derecha de la interfaz gráfica. Resulta necesaria su inclusión puesto que el reproductor de vídeo utilizado no cuenta con una interfaz para el ratón que permita interactuar con la reproducción multimedia. Con la tecla "F1" se puede pausar o despausar el vídeo, con "F2" se retrocederán 5 segundos de vídeo y con la tecla "F3" se avanzarán 5 segundos. La inclusión de estas teclas de utilidad permite poder ver una parte específica del vídeo para comprobar la transcripción, avanzar en el mismo si no se necesita ver cierta parte de la muestra o pausarlo para observar algún patrón específico en el mismo o tomar un descanso; de igual forma, si se despausa el vídeo cuando ya se ha terminado se volverá a poner desde el principio. Se han elegido estas teclas específicas para poder utilizarse mientras se tiene seleccionada la caja de texto donde se encuentra la transcripción, pues además de no modificar el texto de ninguna manera, son fácilmente alcanzables en el teclado.

Capítulo 7

Conclusiones

En este trabajo hemos llevado a cabo el desarrollo de una herramienta para la detección de hablantes en vídeos y la generación de muestras para crear bases de datos, además de la construcción de un modelo de detección del hablante focalizado en la lengua española. Además, se ha realizado la selección de arquitecturas, conjuntos de datos y modelos, incluyendo el entrenamiento y experimentación para asegurar un buen rendimiento. También se han comentado las mejoras realizadas para lidiar con posibles diferencias en datos de entrada más realistas y asegurar la calidad de las muestras generadas.

Tal y como se comentó en la Sección 1.2, nuestros objetivos principales han sido la creación de una herramienta que acelerase en gran medida la anotación de datos y un sistema automatizado para la detección del hablante en español. Hemos alcanzado ambos objetivos y obtenido unos resultados respetables, entrenando y evaluando con el conjunto de datos LIP-RTVE.

A pesar de contar con un conjunto de datos de baja resolución y con cierta cantidad de ruido, creemos que el desempeño del modelo tras su entrenamiento con estos datos representa una calidad más que aceptable, tal y como se ha podido ver en el Capítulo 5.

No obstante, el proyecto no ha estado libre de dificultades y problemas. Un ejemplo de ello es conseguir que las muestras no se terminen abruptamente por una mala clasificación, solucionado gracias al suavizado aplicado tras la clasificación del modelo. Otro gran problema a solucionar ha sido la tendencia del modelo a asignar la misma etiqueta a toda la entrada proporcionada, debido a la naturaleza de los datos empleados en el estudio.

Para concluir, creemos que la herramienta desarrollada para la detección del habla en vídeos puede suponer una contribución valiosa al campo de anotación de datos, ya que permitiría ahorrar muchas horas de trabajo humano. Cabe destacar que a medida que vaya avanzando el estado del arte y las tecnologías empleadas, nuestra herramienta no se quedará atrás, gracias a su capacidad de intercambiar fácilmente el modelo utilizado.

7.1 Trabajos futuros

La detección automática del hablante es una tarea que continúa abierta a la investigación, donde continuamente se presentan nuevos modelos con menor tiempo de inferencia y mejor rendimiento [7, 31]. Dado que se ha creado una herramienta para anotar datos, convendría que tuviese una precisión razonable, con lo que sería de interés también realizar una nueva experimentación con Light-ASD [7] y comparar los resultados. Nos hubiera gustado haber empleado este modelo en nuestras pruebas pero no hemos podido por falta de tiempo, puesto que se ha presentado muy recientemente. Concretamente, en un momento en el que nuestro proyecto ya se encontraba en una etapa avanzada de su desarrollo. Creemos que podría ofrecer mejores resultados y ofrecer un tiempo de inferencia menor, debido a su bajo número de parámetros, tal y como se describía en el Capítulo 2.

Otro gran punto a mejorar sería facilitar la instalación de la herramienta mediante la creación automática de un entorno virtual Python e instalación de dependencias. De esta forma, se proporcionaría una gran sencillez de uso a aquellas personas que no estén familiarizadas en el uso de paquetes en lenguajes de programación, como pueden ser muchos de los anotadores de datos que potencialmente usarían esta herramienta.

La unificación del *script* de detección del habla en vídeos y la interfaz gráfica supondría otra mejora significativa para la herramienta, debido a que actualmente se deben clasificar primero los vídeos para poder posteriormente analizarlos y etiquetarlos. La integración del proceso de detección en la interfaz permitiría al usuario anotador poder visualizar e interactuar con las escenas donde se ha detectado el habla sin tener que realizar un proceso previo.

De igual manera, una importante mejora podría ser la adaptación de la herramienta a una tarea similar a la detección del hablante llamada diarización del hablante [59]. Esta tarea consiste en particionar el vídeo de entrada en segmentos según la identidad de cada hablante, con lo que se podría mejorar substancialmente la calidad de las bases de datos generadas al contar con la identidad de los hablantes, pudiendo así separar las muestras según el hablante al que pertenezcan, un aspecto que permitiría entrenar sistemas dependientes del hablante [60] o discriminar entre ellos mejor [61].

Como última mejora planeada, nos gustaría implementar un etiquetado de muestras más interactivo, de forma que si un usuario encuentra una muestra de un vídeo aceptable pero cree que puede ser mejorada, se podría pulsar un botón. Este botón hará que el vídeo sea re-evaluado posteriormente con otro modelo distinto o con un tamaño de ventana diferente al usado, este cambio podría ser capaz de producir resultados más aptos para ese vídeo concreto.

Bibliografía

[1] Manuel Milling, Florian B Pokorny, Katrin D Bartl-Pokorny y Björn W Schuller. "Is speech the new blood? recent progress in ai-based disease detection from audio in a nutshell". En: *Frontiers in Digital Health* 4 (2022), pág. 886615.

[2] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken y Clara I Sánchez. "A survey on deep learning in medical image analysis". En: *Medical image analysis* 42 (2017), págs. 60-88.

[3] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey e Ilya Sutskever. "Robust speech recognition via large-scale weak supervision". En: *International Conference on Machine Learning*. PMLR. 2023, págs. 28492-28518.

[4] Pingchuan Ma, Stavros Petridis y Maja Pantic. "End-to-end audio-visual speech recognition with conformers". En: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, págs. 7613-7617.

[5] Björn W Schuller y Dagmar M Schuller. "Audiovisual Affect Assessment and Autonomous Automobiles: Applications". En: *arXiv preprint arXiv:2203.07482* (2022).

[6] Abhishek Gupta, Alagan Anpalagan, Ling Guan y Ahmed Shaharyar Khwaja. "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues". En: *Array* 10 (2021), pág. 100057.

[7] Junhua Liao, Haihan Duan, Kanghui Feng, Wanbing Zhao, Yanbing Yang y Liangyin Chen. "A Light Weight Model for Active Speaker Detection". En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, págs. 22932-22941.

[8] Juan León Alcázar, Moritz Cordes, Chen Zhao y Bernard Ghanem. "End-to-end active speaker detection". En: *European Conference on Computer Vision*. Springer. 2022, págs. 126-143.

[9] Junwen Xiong, Yu Zhou, Peng Zhang, Lei Xie, Wei Huang y Yufei Zha. "Look&listen: Multi-modal correlation learning for active speaker detection and speech enhancement". En: *IEEE Transactions on Multimedia* (2022).

[10] Rahul Sharma y Shrikanth Narayanan. "Audio-Visual Activity Guided Cross-Modal Identity Association for Active Speaker Detection". En: *IEEE Open Journal of Signal Processing* 4 (2023), págs. 225-232. DOI: 10.1109/OJSP.2023.3267269.

[11] Julien Besle, Alexandra Fort, C. Delpuech y Marie-Hélène Giard. "Bimodal speech: Early suppressive visual effects in human auditory cortex". En: *The European journal of neuroscience* 20 (nov. de 2004), págs. 2225-34. DOI: 10.1111/j.1460-9568.2004.03670.x.

[12] Chen-Zhao Yang, Jun Ma, Shilin Wang y Alan Wee-Chung Liew. "Preventing deepfake attacks on speaker authentication by dynamic lip movement analysis". En: *IEEE Transactions on Information Forensics and Security* 16 (2020), págs. 1841-1854.

[13] Jan Cech, Ravi Mittal, Antoine Deleforge, Jordi Sanchez-Riera, Xavier Alameda-Pineda y Radu Horaud. "Active-speaker detection and localization with microphones and cameras embedded into a robotic head". En: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. 2013, págs. 203-210. DOI: 10.1109/HUMANOIDS.2013.7029977.

[14] Triantafyllos Afouras, Joon Son Chung y Andrew Zisserman. "LRS3-TED: a large-scale dataset for visual speech recognition". En: *arXiv preprint arXiv:1809.00496* (2018).

[15] Amir Zadeh, Yan Sheng Cao, Simon Hessner, Paul Pu Liang, Soujanya Poria y Louis-Philippe Morency. "CMU-MOSEAS: A multimodal language dataset for Spanish, Portuguese, German and French". En: *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*. Vol. 2020. NIH Public Access. 2020, pág. 1801.

[16] Triantafyllos Afouras, Joon Son Chung, Andrew Senior, Oriol Vinyals y Andrew Zisserman. "Deep audio-visual speech recognition". En: *IEEE transactions on pattern analysis and machine intelligence* 44.12 (2018), págs. 8717-8727.

[17] Pingchuan Ma, Stavros Petridis y Maja Pantic. "Visual speech recognition for multiple languages in the wild". En: *Nature Machine Intelligence* 4.11 (2022), págs. 930-939.

[18] Joseph Roth, Sourish Chaudhuri, Ondrej Klejch, Radhika Marvin, Andrew Gallagher, Liat Kaver, Sharadh Ramaswamy, Arkadiusz Stopczynski, Cordelia Schmid, Zhonghua Xi et al. "Ava active speaker: An audio-visual dataset for active speaker detection". En: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, págs. 4492-4496.

[19] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar et al. "Ava: A video dataset of spatio-temporally localized atomic visual actions". En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, págs. 6047-6056.

[20] Punarjay Chakravarty, Sayeh Mirzaei, Tinne Tuytelaars y Hugo Van hamme. "Who's Speaking? Audio-Supervised Classification of Active Speakers in Video". En: *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ICMI '15. Seattle, Washington, USA: Association for Computing Machinery, 2015, págs. 87-90. ISBN: 9781450339124. DOI: 10.1145/2818346.2820780. URL: https://doi.org/10.1145/2818346.2820780.

[21] Tiago Roxo, Joana C Costa, Pedro RM Inácio y Hugo Proença. "WASD: A Wilder Active Speaker Detection Dataset". En: *arXiv preprint arXiv:2303.05321* (2023).

[22] Pingchuan Ma, Stavros Petridis y Maja Pantic. "Visual speech recognition for multiple languages in the wild". En: *Nature Machine Intelligence* 4.11 (oct. de 2022), págs. 930-939. DOI: 10.1038/s42256-022-00550-z. URL: https://doi.org/10.1038%2Fs42256-022-00550-z.

[23] Helen L Bear, Richard Harvey, Barry-John Theobald y Yuxuan Lan. "Resolution limits on visual speech recognition". En: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2014, págs. 1371-1375.

[24] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu et al. "Conformer: Convolution-augmented Transformer for Speech Recognition". En: *Interspeech 2020* (2020), págs. 5036-5040.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser e Illia Polosukhin. "Attention is all you need". En: *Advances in neural information processing systems* 30 (2017).

[26] Karen Simonyan y Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." En: *ICLR*. Ed. por Yoshua Bengio y Yann LeCun. 2015. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#SimonyanZ14a.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun. "Deep residual learning for image recognition". En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, págs. 770-778.

[28] Maxime Burchi y Radu Timofte. "Audio-visual efficient conformer for robust speech recognition". En: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, págs. 2258-2267.

[29] L Ashok Kumar, D Karthika Renuka, V Harsha Priya y S Sudarshan. "Spoken Language Translation using Conformer model". En: *2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)*. IEEE. 2023, págs. 466-471.

[30] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li y Maosong Sun. "Graph neural networks: A review of methods and applications". En: *AI open* 1 (2020), págs. 57-81.

[31] Kyle Min, Sourya Roy, Subarna Tripathi, Tanaya Guha y Somdeb Majumdar. "Learning long-term spatial-temporal graphs for active speaker detection". En: *European Conference on Computer Vision*. Springer. 2022, págs. 371-387.

[32] Karen Simonyan y Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos". En: *Advances in neural information processing systems* 27 (2014), pág. 568.

[33] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du y Ji-Rong Wen. "Scalable graph neural networks via bidirectional propagation". En: *Advances in neural information processing systems* 33 (2020), págs. 14556-14566.

[34] Mathias Niepert, Mohamed Ahmed y Konstantin Kutzkov. "Learning convolutional neural networks for graphs". En: *International conference on machine learning*. PMLR. 2016, págs. 2014-2023.

[35] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein y Justin M Solomon. "Dynamic graph cnn for learning on point clouds". En: *ACM Transactions on Graphics (tog)* 38.5 (2019), págs. 1-12.

[36] William L Hamilton, Rex Ying y Jure Leskovec. "Inductive representation learning on large graphs". En: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, págs. 1025-1035.

[37] Ruijie Tao, Zexu Pan, Rohan Kumar Das, Xinyuan Qian, Mike Zheng Shou y Haizhou Li. "Is someone speaking? exploring long-term temporal features for audio-visual active speaker detection". En: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, págs. 3927-3935.

[38] Jianbo Guo, Yuxi Li, Weiyao Lin, Yurong Chen y Jianguo Li. "Network Decoupling: From Regular to Depthwise Separable Convolutions". En: *British Machine Vision Conference*. 2018.

[39] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk y Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". En: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, oct. de 2014, págs. 1724-1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179.

[40] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn y Andrew Zisserman. "The Pascal Visual Object Classes (VOC) Challenge." En: *Int. J. Comput. Vis.* 88.2 (2010), págs. 303-338. URL: http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10.

[41] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li y Feiyue Huang. "DSFD: dual shot face detector". En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, págs. 5060-5069.

[42] Shaoqing Ren, Kaiming He, Ross Girshick y Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". En: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 39.06 (2017), págs. 1137-1149.

[43] Fisher Yu y Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". En: *arXiv preprint arXiv:1511.07122* (2015).

[44] Shuo Yang, Ping Luo, Chen-Change Loy y Xiaoou Tang. "Wider face: A face detection benchmark". En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, págs. 5525-5533.

[45] Jie Hu, Li Shen y Gang Sun. "Squeeze-and-excitation networks". En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, págs. 7132-7141.

[46] Sep. de 2022. URL: https://openai.com/research/whisper.

[47] David Gimeno-Gómez y Carlos-D. Martínez-Hinarejos. "LIP-RTVE: An Audiovisual Database for Continuous Spanish in the Wild". En: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, jun. de 2022, págs. 2750-2758. URL: https://aclanthology.org/2022.lrec-1.294.

[48] George Saon, Hagen Soltau, David Nahamoo y Michael Picheny. "Speaker adaptation of neural network acoustic models using i-vectors". En: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE. 2013, págs. 55-59.

[49] Felix Weninger, Jesús Andrés-Ferrer, Xinwei Li y Puming Zhan. "Listen, Attend, Spell and Adapt: Speaker Adapted Sequence-to-Sequence ASR". En: *Proc. Interspeech 2019* (2019), págs. 3805-3809.

[50] David Gimeno-Gómez y CD Martínez-Hinarejos. "Speaker-Adapted End-to-End Visual Speech Recognition for Continuous Spanish". En: *Proceedings of the IberSPEECH, Granada, Spain* (2022), págs. 14-16.

[51] Mark Gales, Steve Young et al. "The application of hidden Markov models in speech recognition". En: *Foundations and Trends® in Signal Processing* 1.3 (2008), págs. 195-304.

[52] Jason Brownlee. *A gentle introduction to threshold-moving for Imbalanced Classification*. Ene. de 2021. URL: `https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/`.

[53] Leo Liberti, Carlile Lavor, Nelson Maculan y Antonio Mucherino. "Euclidean distance geometry and applications". En: *SIAM review* 56.1 (2014), págs. 3-69.

[54] Shuang Yang, Yuanhang Zhang, Dalu Feng, Mingmin Yang, Chenhao Wang, Jingyun Xiao, Keyu Long, Shiguang Shan y Xilin Chen. "LRW-1000: A naturally-distributed large-scale benchmark for lip reading in the wild". En: *2019 14th IEEE international conference on automatic face & gesture recognition (FG 2019)*. IEEE. 2019, págs. 1-8.

[55] Joon Son Chung y Andrew Zisserman. "Lip reading in the wild". En: *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II 13*. Springer. 2017, págs. 87-103.

[56] Suramya Tomar. "Converting video formats with FFmpeg". En: *Linux Journal* 2006.146 (2006), pág. 10.

[57] Md Rashidul Hasan, Mustafa Jamil, MGRMS Rahman et al. "Speaker identification using mel frequency cepstral coefficients". En: *variations* 1.4 (2004), págs. 565-568.

[58] VideoLan. *VLC media player*. 2006. URL: `https://www.videolan.org/vlc/index.html`.

[59] Sue E Tranter y Douglas A Reynolds. "An overview of automatic speaker diarization systems". En: *IEEE Transactions on audio, speech, and language processing* 14.5 (2006), págs. 1557-1565.

[60] Xuedong Huang y Kai-Fu Lee. "On speaker-independent, speaker-dependent, and speaker-adaptive speech recognition". En: *IEEE Transactions on Speech and Audio processing* 1.2 (1993), págs. 150-157.

[61] Arsha Nagrani, Joon Son Chung y Andrew Zisserman. “Voxceleb: a large-scale speaker identification dataset”. En: *arXiv preprint arXiv:1706.08612* (2017).