ISSN 1688-2806

Universidad de la República
Facultad de Ingeniería

Herramientas computacionales para el análisis del entorno sonoro urbano

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Pablo Zinemanas

en cumplimiento parcial de los requerimientos para la obtención del título de Magister en Ingeniería Eléctrica.

Directores de Tesis

Pablo Cancela ..... Universidad de la República
Alice Elizabeth González ..... Universidad de la República

Tribunal

Mauricio Delbracio ..... Universidad de la República
Leonardo de Oliveira Nunes ..... Microsoft
Justin Salamon ..... Adobe Research

Director Académico

Pablo Cancela ..... Universidad de la República

Montevideo
lunes 13 mayo, 2019

Herramientas computacionales para el análisis del entorno sonoro urbano, Pablo Zinemanas.

ISSN 1688-2806

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 109 páginas.
Compilada el lunes 13 mayo, 2019.
http://iie.fing.edu.uy/

Sanlúcar de Barrameda, España.
Fotografía: Ana Márquez

Agradecimientos

Agradezco a Martín Rocamora y Pablo Cancela por su apoyo, dedicación y acompañamiento durante todo el trabajo de esta tesis. También agradezco a Ignacio Irigaray y al resto de compañeros del Instituto de Ingeniería Eléctrica que fueron parte de esta etapa.

Quiero agradecer a mi familia por acompañarme y guiarme en todas las decisiones y caminos de la vida.

Agradezco profundamente a mis amigos, en especial a Maite y Claudia, por su apoyo constante en esta y tantas otras etapas de mi vida. También a Martín por compartir tantas charlas de *café*.

Finalmente quiero agradecer muy especialmente a Ana, por su cariño y apoyo incondicional en todos los momentos.

A Gela.

Resumen

Muchos centros urbanos presentan niveles de ruido que pueden ser molestos para sus habitantes, o incluso nocivos para la salud. El ruido puede interferir en la vida cotidiana modificando la conducta de las personas y afectando el descanso. La exposición prolongada a niveles de intensidad de sonido elevados produce daños en la salud. Por esta razón el entorno sonoro constituye frecuentemente una problemática en las ciudades, que cada vez tiene mayor presencia en los reclamos de las comunidades y grupos ambientalistas, en la planificación gubernamental y en los estudios académicos.

En los últimos años ha surgido el interés por desarrollar tecnologías para el monitoreo y diagnóstico del entorno sonoro urbano, orientadas a facilitar la planificación y la gestión de la ciudad. Se basan en una red de sensores distribuidos que permiten registrar audio y estimar los niveles de ruido en tiempo real. Además, mediante el uso de tecnologías de procesamiento de señales y aprendizaje automático, se busca generar de forma automática una descripción del tipo de ambiente sonoro registrado, incluyendo las fuentes que lo componen. Esto permite a las autoridades aplicar medidas correctivas, o desarrollar planes de monitoreo y gestión usando información del entorno sonoro.

En esta tesis se estudian herramientas computacionales para la caracterización de entornos sonoros urbanos. El foco del trabajo es la detección de eventos sonoros, lo que implica la clasificación de las fuentes presentes en el registro sonoro incluyendo su tiempo de inicio y duración. Se relevan las bases de datos disponibles y se concluye que es necesario crear propios de la ciudad de Montevideo. Esto conduce a la creación de la base de datos SonidosMVD, para lo que es preciso definir las clases de sonidos de interés y una taxonomía que las ordena. Por otro lado, se estudian las medidas de desempeño más utilizadas para este problema y se señalan los inconvenientes que surgen cuando se tiene un conjunto de datos muy desbalanceado. Se proponen nuevas medidas de desempeño que buscan ser mas adecuadas para este problema.

Se presentan experimentos de detección de eventos sonoros con distintos algoritmos de clasificación sobre distintos conjuntos de datos. En primer lugar se utilizan técnicas que involucran el procesamiento de señales para la extracción de características y reconocimiento de patrones para la clasificación. Luego, se presentan técnicas de aprendizaje profundo. Se diseña un modelo *end–to–end*, cuya entrada es la forma de onda de la señal y la salida es el vector de clasificación, y se muestra su utilidad para este problema. Esto se realiza concatenando dos redes:

una para la extracción de características y otra para la clasificación. Se muestra que se pueden diseñar dichos modelos utilizando conocimiento sobre el problema para comenzar el entrenamiento desde un punto inicial que ha demostrado brindar buenos resultados. Con este sistema se obtienen resultados de clasificación similares a los del estado el arte mientras que se disminuye el número de parámetros involucrados.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Motivación y antecedentes

El entorno sonoro en el que están inmersos los habitantes de una ciudad es reflejo de sus costumbres y modos de vida, y a la vez los condiciona profundamente [1]. En la actualidad, muchos centros urbanos presentan niveles de ruido que pueden ser molestos para sus habitantes, o incluso nocivos para la salud. El ruido puede interferir en la vida cotidiana modificando la conducta de las personas y afectando el descanso. La exposición prolongada a niveles de intensidad de sonido elevados produce daños en la salud, como por ejemplo, pérdida de audición, aparición de zumbidos (tinitus), trastornos del sueño o estrés [2]. Por esta razón el entorno sonoro constituye frecuentemente una problemática en las ciudades, que cada vez tiene mayor presencia en los reclamos de las comunidades y grupos ambientalistas, en la planificación gubernamental y en los estudios académicos [3]. En particular en Montevideo, los reclamos por contaminación acústica ocupan el segundo lugar en números generales entre todos los recibidos por la Defensoría del Vecino en 2015, 2016 y 2017 [4–6].

Uno de los principales antecedentes del estudio sistemático del entorno sonoro es el trabajo de Murray Schafer, quien a fines de la década de 1960 dirige el proyecto *The World Soundscape* con un grupo de investigadores de la Universidad Simon Fraser de Vancouver [7]. Del trabajo de este grupo surge el concepto de *paisaje sonoro* (*soundscape*) que se presenta en el libro *The New Soundscape* [8]. Schafer usó los términos paisaje sonoro y ecología acústica para describir críticamente nuestro medio ambiente. En este marco se desarrolla en nuestro país desde el año 2000 el proyecto *Paisaje Sonoro Uruguay* [9], en el Estudio de Música Electroacústica de la Escuela Universitaria de Música de la Universidad de la República (EUM). Tiene como objetivo principal estudiar el paisaje sonoro uruguayo con énfasis en el registro, la creación artística y actividades de sensibilización. Además, dispone de un archivo en línea de registros geolocalizados [10].

A su vez, en nuestro país se estudia el entorno sonoro en el Departamento de Ingeniería Ambiental del Instituto de Mecánica de Fluidos e Ingeniería Ambiental (IMFIA). Una línea de investigación de este departamento es la de mapas acústicos

y redes de monitoreo de ruido urbano [11]. En 1999 en el marco de un convenio entre el IMFIA y la Intendencia de Montevideo (IM) se realizó un estudio del mapa acústico de la ciudad. El resultado de dicho trabajo se presenta en un informe final que detalla la metodología utilizada y los Niveles de Presión Sonora (SPL) en diferentes zonas de la ciudad y en distintos momentos [12].

En los últimos años ha surgido el interés por desarrollar tecnologías para el monitoreo y diagnóstico del entorno sonoro urbano, orientadas a facilitar la planificación y la gestión de la ciudad [3,13]. Se basan en una red de sensores distribuidos que permiten registrar audio y estimar los niveles de ruido en tiempo real [14,15]. Además, mediante el uso de tecnologías de procesamiento de señales y aprendizaje automático, se busca generar de forma automática una descripción del tipo de ambiente sonoro registrado, incluyendo las fuentes que lo componen [16–18]. Esto permite a las autoridades aplicar medidas correctivas, o desarrollar planes de monitoreo y gestión usando información del entorno sonoro.

Dentro de las iniciativas más destacadas, el proyecto Sounds of New York City (SONYC) [13], iniciado en 2014, busca desarrollar un sistema de monitoreo de entorno sonoro urbano. En este proyecto se ha llevado adelante el estudio de diversos aspectos del problema, como el diseño de sensores [14], la elección de micrófonos adecuados [15], la clasificación automática [16,18] y el diseño de bases de datos de registros del entorno sonoro urbano, para entrenamiento y validación de resultados [17,19].

Una de las primeras reuniones académicas orientadas específicamente a la detección y clasificación automática de escenas acústicas y eventos sonoros es CLEAR que tuvo lugar en 2007 [20]. La primera edición del *Workshop on Detection and Classification of Acoustic Scenes and Events* - DCASE se realiza en 2013 y evidencia el interés que ha despertado el tema en la comunidad científica [21]. En ambos encuentros, se realizan contiendas de algoritmos sobre bases de datos que no incluyen entorno sonoro urbano. En 2016 se realiza la primera contienda con grabaciones de audio generadas en entornos residenciales (suburbanos), Challenge DCASE 2016, que involucra varios problemas de análisis automático de entorno sonoro, como clasificación de ambientes y detección de eventos. Se busca que esta iniciativa, permita evaluar distintos enfoques propuestos por los diferentes grupos de investigación utilizando una misma base de datos de acceso público [22] y las mismas medidas de desempeño [23], para fomentar la reproducibilidad de los resultados y su adecuada comparación.

1.2. Definición del problema y alcance

Este trabajo se centra en el estudio de las herramientas computacionales para la caracterización de entornos sonoros urbanos, en particular, se enfoca en la clasificación y detección de eventos sonoros. Estos dos problemas se ilustran en el diagrama de la Figura 1.1, a partir de sistemas que extraen información sobre las clases de sonidos presentes en una señal de audio.

Se llama *clasificación* al problema de categorizar un registro sonoro (o fragmento de audio de cierta duración) con etiquetas de clases predefinidas [24]. En este

caso cada registro de audio es clasificado según la clase predominante que el sistema logra reconocer. Este problema es interesante como forma de acercamiento a la caracterización de entornos sonoros urbanos pero es insuficiente para describirlos, ya que suelen tener varias fuentes en simultáneo.

Por lo anterior, se busca diseñar sistemas más complejos para la caracterización de los registros. Una opción en este sentido es la detección de eventos sonoros, que refiere al problema de encontrar todas las fuentes presentes en un registro de audio incluyendo la información de comienzo y final de cada evento [24] como se ilustra en la Figura 1.1.

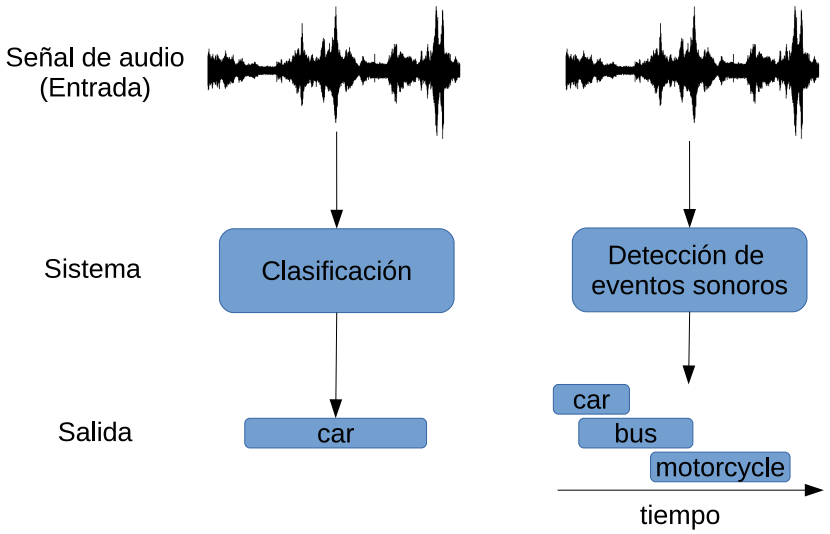


Figura 1.1: Diagrama de comparación de sistemas de clasificación de registros sonoros (izquierda) y detección de eventos sonoros (derecha).

El estudio de estos problemas en un entorno urbano es muy desafiante debido a sus características acústicas inherentes. En primer lugar, en una misma clase de sonidos se pueden tener características muy diferentes; por ejemplo, se puede pensar en un auto antiguo y uno moderno, ambos podrían pertenecer a la misma clase, pero claramente producen sonidos diferentes. Otro aspecto desafiante es el solapamiento temporal de eventos que evidentemente existe en los entornos urbanos y dificulta la clasificación. Además, las grabaciones de este tipo de entorno suelen hacerse desde un dispositivo de grabación alejado de la fuente, lo que aumenta la variación de la transferencia entre la fuente y el micrófono [24].

Por otro lado, existen desafíos vinculados a los datos (registros sonoros) que se utilizan para entrenar los sistemas de detección y clasificación. Por un lado, las bases de datos disponibles con registros de entornos urbanos son pocas y su tamaño es relativamente pequeño [17,19,22]. Esto dificula la aplicación de técnicas de aprendizaje profundo (*deep learning*), ya que requieren grandes cantidades de datos para entrenamiento. Además, estas bases de datos están diseñadas en función de ciertas clases de interés y cómo estas se relacionan. Por ejemplo en [17] se define una organización taxonómica de clases en entornos urbanos a partir de un estudio

sobre denuncias de ruidos molestos; mientras que en [22] las grabaciones se realizan en entornos residenciales, por lo que algunas de las clases definidas en un caso no se aplican necesariamente al otro. De este modo, en la comunidad científica no hay un consenso sobre la taxonomía u ontología que se debe utilizar para este tipo de entornos sonoros [17, 24, 25]. Por lo tanto, en este trabajo se considera necesario definir una forma de organizar las clases para los tipos de sonidos de interés. Una vez definidas las clases de interés y cómo se ordenan, se diseña una base de datos con registros sonoros de la ciudad de Montevideo de forma de realizar pruebas en condiciones reales.

1.3. Objetivos

El objetivo principal de este trabajo es estudiar los algoritmos del estado del arte en detección de eventos sonoros y clasificación de fuentes aplicados a grabaciones del entorno sonoro urbano. Se espera contribuir a la caracterización automática de entornos sonoros urbanos abordando el problema en condiciones reales. Además, se pretende estudiar técnicas recientes de clasificación, buscando identificar oportunidades de mejora y proponer soluciones novedosas.

Los objetivos específicos de este trabajo son los siguientes:

- Estudiar e implementar los principales algoritmos del estado del arte en detección y clasificación de eventos sonoros en entornos sonoros urbanos. Esto incluye estudiar técnicas de aprendizaje profundo (*deep learning*), haciendo énfasis en la interpretabilidad de su funcionamiento y buscando proponer soluciones novedosas.
- Diseñar experimentos para evaluar el resultado de dichos algoritmos.
- Estudiar las bases de datos disponibles de registros sonoros en entornos urbanos y hacer un análisis sobre sus características.
- Diseñar y construir una base de datos de registros sonoros de Montevideo que permita el estudio de los algoritmos desarrollados en condiciones reales con las características propias de esta ciudad. Esto incluye el diseño de una taxonomía que permita definir y ordenar las clases de interés.
- Utilizar una metodología de trabajo que evite sesgos en la evaluación de resultados. En este sentido, se busca estudiar las medidas de desempeño utilizadas en el estado del arte, buscando comprender sus posibles sesgos.

1.4. Organización del documento

Para facilitar su lectura, el documento se centra en el problema de detección de eventos sonoros. El trabajo realizado durante la tesis sobre clasificación de registros sonoros se presenta en el apéndice A. El resto del documento se organiza de la siguiente forma.

Capítulo 2. Base de datos. Se presentan las bases de datos de registros sonoros urbanos disponibles y en particular, se describen aquellas que fueron utilizadas en este trabajo. Además, se describe el diseño y el proceso de creación de una base de datos de registros sonoros de la ciudad de Montevideo, con especial énfasis en los aspectos metodológicos.

Capítulo 3. Evaluación y medidas de desempeño. Se describe la metodología de evaluación de los sistemas desarrollados y se presentan las medidas de desempeño utilizadas.

Capítulo 4. Extracción de características Se exponen las características de audio que se utilizan en todos los algoritmos desarrollados durante el trabajo, incluyendo los procesos de integración temporal y normalización.

Capítulo 5. Detección basada en aprendizaje poco profundo (shallow learning). Se aborda la etapa de clasificación en el problema de detección de eventos sonoros. Se aplican técnicas de procesamiento de señales y reconocimiento de patrones y se replican los resultados de varios trabajos del estado del arte.

Capítulo 6. Detección basada en aprendizaje profundo (deep learning). Se presenta una breve introducción de los principales modelos de *deep learning* utilizados para el problema de detección de eventos sonoros. Se muestran resultados de modelos del estado del arte y se estudia un enfoque novedoso en base a arquitecturas punta a punta (*end–to–end*).

Capítulo 7. Conclusiones. Se presentan las conclusiones y algunas perspectivas para trabajo futuro.

Capítulo 2

Bases de datos

> *"Considera, por ejemplo, los procesos que llamamos «juego». Me refiero a juegos de tablero, juegos de cartas, juegos de pelota, juegos de lucha, etcétera. ¿Qué hay de común a todos ellos?... Podemos ver como los parecidos surgen y desaparecen. Y el resultado de este examen reza así: vemos una complicada red de parecidos que se superponen y entrecruzan. Parecidos a gran escala y de detalle."*
>
> – Ludwig Wittgenstein, Investigaciones Filosóficas

En este capítulo se presentan las bases de datos de registros sonoros en ambientes urbanos que son utilizadas en este trabajo y otras que se encuentran disponibles. Luego se expone el proceso de creación de una base de datos con registros de la ciudad de Montevideo. Esto incluye, las discusiones sobre la taxonomía utilizada, el diseño de la base y los procesos de grabación y etiquetado, con especial énfasis en los aspectos metodológicos.

2.1. Bases de datos disponibles

Aunque existen varias bases de datos que contienen registros sonoros ambientales[1], este trabajo se centra en aquellas con foco en los entornos urbanos. En el Capítulo 5 y el Apéndice A se presentan resultados con dos bases de datos de este tipo: *TUT database* [22] y *UrbanSound8k* [17]. En el Capítulo 6 se muestran los resultados de aplicar modelos de aprendizaje profundo entrenados con la base de datos *URBAN-SED* [19].

2.1.1. UrbanSound8k

En el marco del proyecto SONYC, que busca monitorear el entorno sonoro de la ciudad de New York [27], se ha desarrollado la base de datos *UrbanSound8k* [17]. Esta base cuenta con más de ocho mil archivos de audio y cada uno de ellos está etiquetado como perteneciente a una de las siguientes clases:

[1] En [26] se puede encontrar un listado completo de bases de datos a agosto de 2016.

- Air conditioner
- Car horn
- Children playing
- Dog bark
- Drilling
- Engine idling
- Gun shot
- Jackhammer
- Police siren
- Street music

Ha sido construida en base a los registros sonoros disponibles en la plataforma *freesound*[2]. Para cada una de las clases, se descargaron todos los archivos disponibles en la plataforma pertenecientes a dicha clase y se hizo una revisión manual, con el objetivo de mantener sólo aquellos que tuvieran alguna fuente sonora de dicho tipo. Luego, los archivos seleccionados fueron divididos en 8732 fragmentos de máximo 4 segundos de duración, totalizando 8.75 horas de grabación. A su vez, los archivos fueron divididos en 10 particiones del mismo tamaño (*folds*). La base de datos se pone a disposición particionada de esta forma, para facilitar la comparación de desempeño de algoritmos evaluados con validación cruzada. Como los archivos de la base son descargados de *freesound*, la tasa de muestreo y la profundidad de bit no son iguales para todos los registros.

Esta base datos tiene como ventaja que es relativamente grande, pero también tiene varios desafíos. Por ejemplo, hay algunos archivos muy cortos, de menos de un segundo. También hay archivos con etiqueta incorrecta (o por lo menos dudosa), mientras que en otros casos, la fuente marcada se encuentra en el fondo sonoro junto con mucho ruido ambiente. Por otro lado, los archivos están etiquetados con una sola clase, aunque en algunos casos hay varias fuentes sonoras simultáneas. Por lo tanto, la base de datos *UrbanSound8k* no es apropiada para la detección de eventos sonoros que pueden ocurrir de forma simultánea.

2.1.2. URBAN-SED

URBAN-SED, de reciente publicación, es una base de datos para la detección de eventos sonoros basada en la *UrbanSound8k*. Esta base fue generada con la biblioteca para síntesis de entornos sonoros *scaper* e incluye 10000 archivos [19]. Cada archivo de audio se genera sobre una base de ruido *browniano* y se agregan eventos sonoros pertenecientes a los registros de *UrbanSound8k*. Además, a cada

[2]https://freesound.org

uno de los eventos se le aplica pequeñas transformaciones de variación de tono, velocidad e intensidad. Como esta base se utiliza para la detección de eventos, las etiquetas no sólo indican los sonidos presentes sino también el momento de ocurrencia. La base totaliza 30 horas de audio e incluye unos 50000 eventos sonoros. En la Figura 2.1 se muestra la duración total de los eventos para cada una de las clases; notar que existe un buen equilibrio entre las distintas clases. Aunque esta base es útil para probar sistemas de detección de eventos sonoros con redes neuronales (ver Capítulo 6), por construcción cada archivo de audio de la *UrbanSound8k* utilizado puede no corresponder únicamente a una fuente sonora.

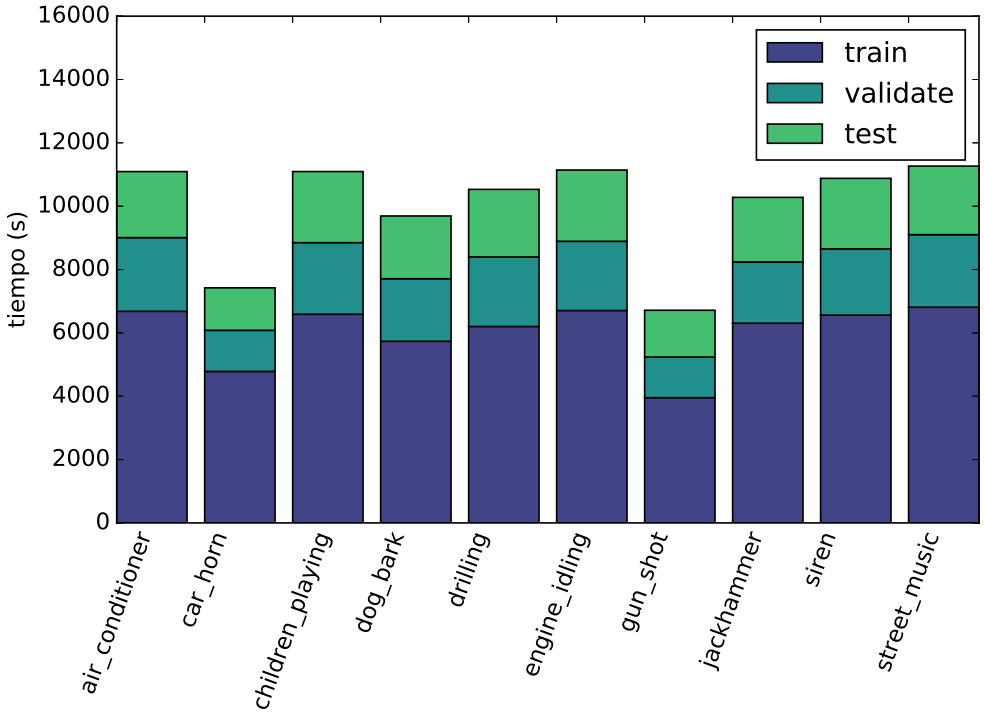


Figura 2.1: Duración total de eventos para cada clase en los conjuntos de *train*, *validate* y *test* en la base de datos URBAN-SED.

2.1.3. TUT database

Otra base de datos disponible para entrenar sistemas de detección de eventos sonoros es la *TUT database* que cuenta con 17 grabaciones, totalizando una hora de duración [22]. Esta base fue dividida de tal forma que 12 grabaciones son utilizadas para el entrenamiento y las 5 restantes sirvan para la evaluación. Además, los datos para entrenamiento están divididos en 4 *folds* para estimar el desempeño del sistema con validación cruzada. Las datos fueron capturados con una grabadora Roland Edirol R09 a una frecuencia de muestreo de 44.1 kHz y una resolución de

24 bits [22]. Las clases etiquetadas en este registro de datos y la duración total de los eventos se puede ver en la Figura 2.2; notar que la clases se encuentran mucho más desbalanceadas que en la base URBAN-SED.

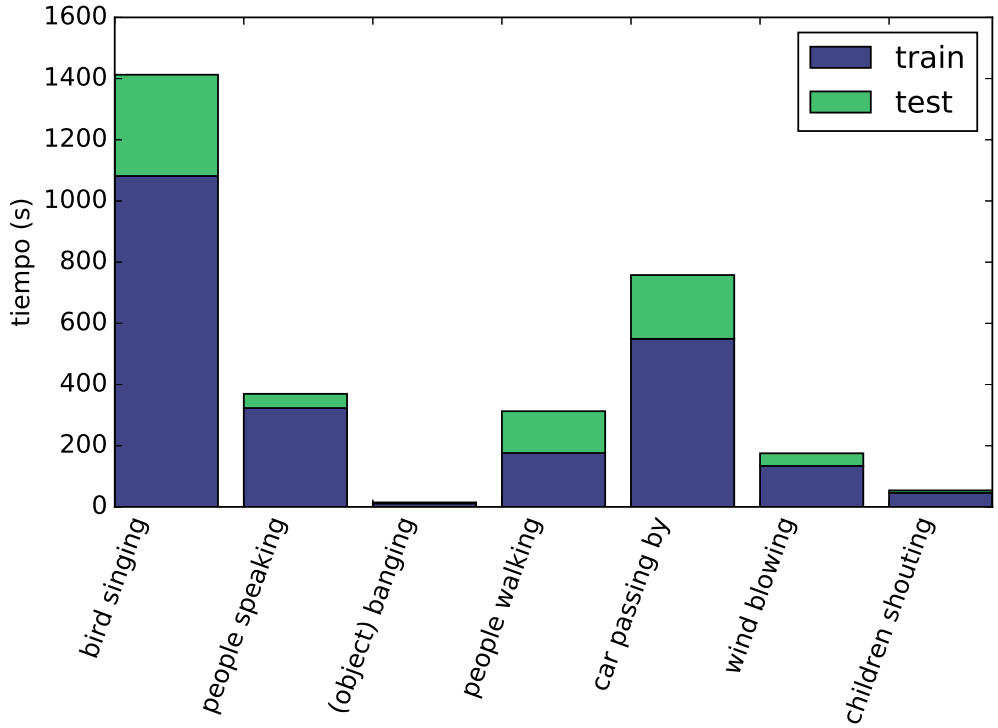


Figura 2.2: Duración total de eventos para cada clase en los conjuntos de *train* y *test* en la base de datos TUT.

Una de las limitantes de esta base es que es relativamente chica ya que está conformada por una hora de grabación. Además, como se ve en el Capítulo 5, presenta errores en el etiquetado, lo que limita el desempeño de los algoritmos que la utilizan.

Además, las clases de interés y la estructura de *TUT database* difieren de lo que se busca en ese trabajo. En particular, al ser una base generada en entornos residenciales, algunas de las clases que han sido etiquetadas no pueden aplicarse en un entorno urbano más denso. Por otro lado, el foco de este trabajo está puesto en los sonidos que pueden considerarse molestos para las personas. El concepto de sonido molesto es subjetivo por lo que las clases deben definirse explícitamente. A priori, algunas clases de sonidos de *TUT database* difícilmente puedan ser consideradas como molestas (p. ej. BIRD SINGING, PEOPLE WALKING o WIND BLOWING). En la siguiente sección se profundiza el estudio de las clases de interés y sus relaciones; se relevan las propuestos en trabajos anteriores y se define el criterio que se utiliza en este trabajo para diseñar la base de datos.

2.1.4. Otras bases de datos

Existen otras bases de datos disponibles que, si bien no han sido utilizadas en este trabajo, se indican a continuación. *Environmental Sound Classification* (ESC) es una base de datos de construcción similar a la UrbanSound8k [28]. También se compone de registros sonoros que pertenecen a una sola clase y fueron obtenidos de la plataforma *freesound*. ESC-50 es una versión de esta base de datos con 50 clases de sonidos que se agrupan en cinco grandes categorías: animales, naturales, humanos, domésticos y exteriores/urbanos. Dentro de esta última categoría solo algunas de clases son de interés para este trabajo.

Otra base de datos que puede ser útil, principalmente para entrenar modelos basados en redes neuronales es la AudioSet, que incluye archivos de audio de 10 segundos que pueden pertenecer a 527 clases de sonidos y totalizan 5800 horas de audio [29]. A pesar de su gran tamaño, solo algunas de las clases son de interés para este trabajo. Por otro lado, la base se generó utilizando videos de *YouTube* y las etiquetas no fueron verificadas en todos los casos. Además la base de datos es similar a la *UrbanSound8k* en el sentido de que las etiquetas no dan cuenta de la presencia de eventos sonoros simultáneos.

2.2. Base de datos SonidosMVD

De lo expuesto en la sección anterior, surge la necesidad de generar datos para el desarrollo de esta tesis. En particular, de las bases de datos disponibles para detección de eventos sonoros y clasificación, algunas son pequeñas, y otras, a pesar de su de gran tamaño, tienen etiquetas que no son de interés para el problema planteado. Además, durante este trabajo se pudo constatar que existen errores de etiquetado (omisiones, sustituciones, y falta de precisión en el intervalo temporal), lo que dificulta el desarrollo de algoritmos de aprendizaje automático. En particular hace muy difícil realizar una correcta evaluación de los resultados, de los casos de éxito y de los casos de error.

2.2.1. Diseño

Antes de diseñar una base de datos, el primer paso es definir las clases de interés y cómo estas se relacionan en función del problema que se quiere atacar. En el proyecto SONYC, los autores proponen una categorización taxonómica de sonidos urbanos. En el nivel superior, se definen cuatro grupos: humano, natural, mecánico y musical, que han sido utilizados en trabajos anteriores. Para definir los niveles inferiores se analizan los reclamos por ruidos molestos en la ciudad de New York de 2010 a 2014 [17]. Por otro lado, en la base de datos TUT se definen las clases durante el proceso de etiquetado, como se explica a continuación. Primero se pide a los participantes que marquen todos los eventos sonoros de forma libre, y posteriormente las etiquetas se agrupan en conceptos más generales. Además, se busca que las etiquetas sean compuestas por un sustantivo y un verbo, como por ejemplo ENGINE ACCELERATING [22, 30].

Este trabajo también se centra en el problema de los ruidos molestos, en particular en el tránsito, ya que este es el origen más frecuente a nivel mundial de la contaminación sonora, generando el mayor daño a la salud de las personas [2]. De esta forma, la taxonomía es definida para los eventos presentes en escenas acústicas donde el tránsito tiene un papel predominante. Se podría ampliar esta taxonomía para que incluya otras fuentes de ruidos molestos como aquellas relacionadas con el ruido social, el ruido de la construcción, el ruido de fábricas y talleres.

Una vez definido el ruido de tránsito como foco de este trabajo, surge la necesidad de definir las clases de interés. Los vehículos (automóviles, ómnibus, motocicletas y camiones) son las principales fuentes de ruido en el tránsito. En algunos casos es difícil definir de qué tipo de vehículo se trata: por ejemplo los denominados vehículos utilitarios pueden considerarse tanto automóviles como pequeños camiones, y hasta podrían considerarse como una clase independiente.

Además, los vehículos generan distintos tipos de sonidos, por ejemplo el sistema de frenos, la rodadura o el motor. Por lo tanto, es necesario que la clasificación de la fuente sonora sea más específica que la del tipo de vehículos. Una forma de acercamiento a este problema es clasificar los eventos sonoros con diferentes atributos correlacionados, como la fuente sonora (objeto), la acción y el contexto [25]. Además, estos atributos pueden estar definidos por una o varias taxonomías u ontologías. Esto implica que no es necesario que haya un único modo de categorizar los eventos. Un mismo evento puede ser clasificado por varios esquemas en simultáneo [25]. En este caso, el contexto ya está definido por los entornos urbanos con predominancia del tránsito. Las fuentes sonoras y las acciones se pueden describir por varias categorías, por ejemplo unas que definen el tipo de vehículo y otras que definen el componente interno que genera el sonido.

Entonces, para este trabajo se define una taxonomía basada en un grafo como el de la Figura 2.3. Se pueden diferenciar dos ramas que se combinan en el medio: una superior que describe las categorías de vehículos; y una inferior que describe las categorías de componentes. Las categorías señaladas en negrita son las que se denominan nivel básico (CAR, BUS, etc. para el caso de vehículos y ENGINE, WHEEL, etc. para el caso de componentes). Las dos ramas de la taxonomía se fusionan en lo que se denomina nivel subordinado (en cursiva) que incluye combinaciones de categorías de vehículos y componentes, con el objetivo de formar etiquetas con mayor detalle sobre la fuente de ruido (CAR/ENGINE IDLING, BUS/COMPRESSOR). Notar que las categorías de componentes se combinan además para formar un par objeto-acción. Esto también se podría hacer en la rama superior para las categorías de vehículos, por ejemplo BUS PASSING BY, CAR STOPPING, etc., aunque para este trabajo parece un poco redundante. Notar que en la Figura 2.3 no están listadas todas las clases que se etiquetaron, es solo un diagrama ilustrativo.

2.2.2. Grabación

A continuación se describe el proceso de generación de la base datos SonidosMVD. Se eligen cuatro ubicaciones en la ciudad de Montevideo para realizar las grabaciones, las que se indican en el mapa de la Figura 2.4. Notar que las

locaciones tienen características de tránsito y uso social diferentes:

L1. Av. Rivera entre 14 de julio y 2 de mayo: zona residencial, con comercios y muchas líneas de ómnibus en ambos sentidos.

L2. Plaza Templete: frente al Parque Rodó, zona de distensión con fuerte presencia de aves, sin residencias ni comercios, con leve flujo de ómnibus.

L3. Acevedo Díaz y San Salvador (frente a plaza Eduardo Acevedo hijo): aunque cercana a la anterior, tiene mucho más presencia de tránsito y residencias, y menos naturaleza.

L4. Magallanes entre San Salvador e Isla de Flores: zona residencial con pocos comercios y pocas líneas de ómnibus.

Las registros sonoros se generan con una grabadora SONY PCM-D50 a una tasa de muestreo de 48 kHz y una resolución de 24 bits. También se graba video con una cámara GOPRO Hero 3 a una tasa de 30 frames por segundo y resolución de 1920×1080 píxeles. En la Figura 2.5 se puede ver la configuración de los equipos de grabación en una de las locaciones (L2). Se grabaron archivos de audio y video

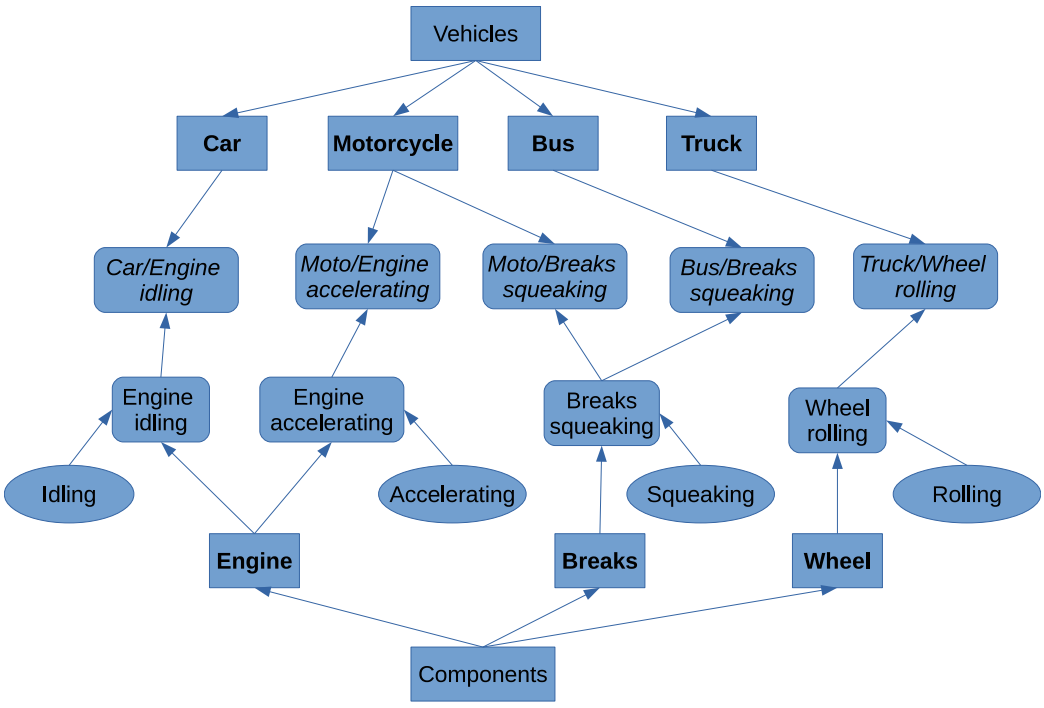


Figura 2.3: Taxonomía definida en forma de grafo. En la rama superior se encuentran las categorías de vehículos y en la inferior las de componentes. Los niveles básicos de etiquetas se marcan con negrita, mientras que el nivel subordinado en cursiva. Los bloques rectangulares describen objetos; las elipses, acciones; y los rectángulos redondeados, combinaciones objeto-acción.

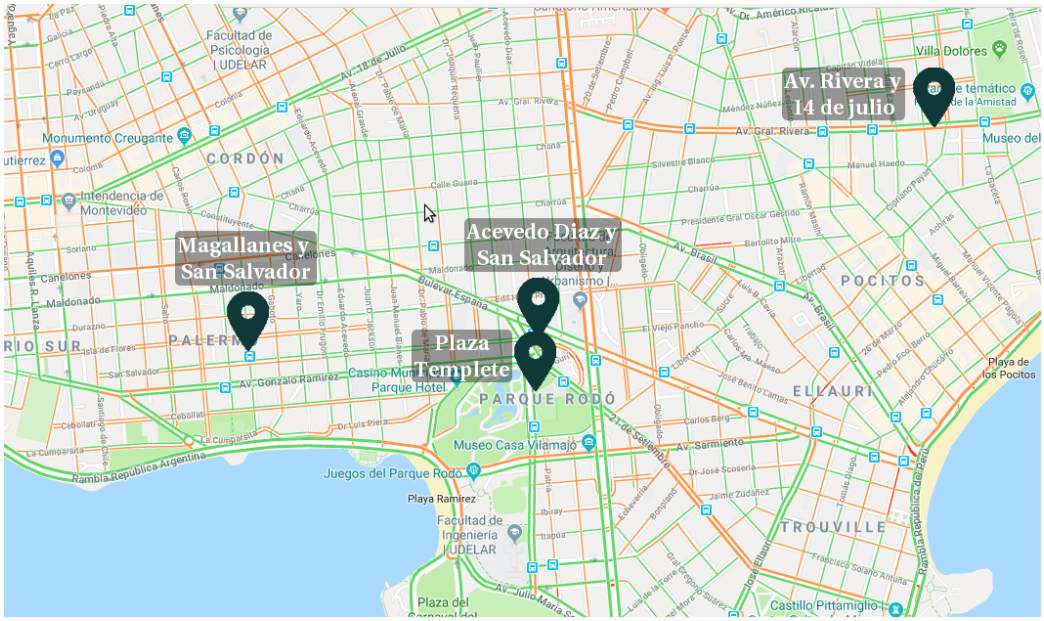


Figura 2.4: Ubicaciones usadas para la generación de la base de datos. El mapa, extraído de Google Maps, representa en colores la información de niveles de tráfico promedio para un lunes a las 9:00 según tres categorías: verde, naranja y rojo (de menor a mayor intensidad).

de unos 15 minutos en diferentes horarios en las distintas locaciones. El video es registrado ya que resulta de mucha utilidad para el proceso de etiquetado.

El procesamiento de los archivos incluyó la sincronización del audio y el video, la eliminación de períodos con mucho viento y la segmentación en archivos de aproximadamente cinco minutos para facilitar su manipulación. Los conjuntos de train y validate se componen de 24 y 7 archivos de la locación L1 respectivamente, mientras que el conjunto de test se compone de 16 archivos de las locaciones L2, L3 y L4. La base totaliza 233 minutos, 117 del conjunto train, 33 de validate y 83 de test.

Figura 2.5: Configuración de los equipos de grabación en una de las locaciones (L2).

2.2.3. Etiquetado

Para etiquetar los registros de la base de datos se utiliza el programa ELAN [31]. Este software permite etiquetar un archivo de audio, mientras que se accede a otras formas de visualización disponibles en formato video. Esta función se utiliza para poder ver el archivo de video de la grabación y el espectrograma de la señal. Para esto último, se generaron archivos de video con el espectrograma de la señal de audio y una línea que marca el instante temporal (ver Figura 2.6). Otra característica interesante del software elegido es que permite generar etiquetas jerárquicas, lo que es muy útil para la taxonomía definida anteriormente.

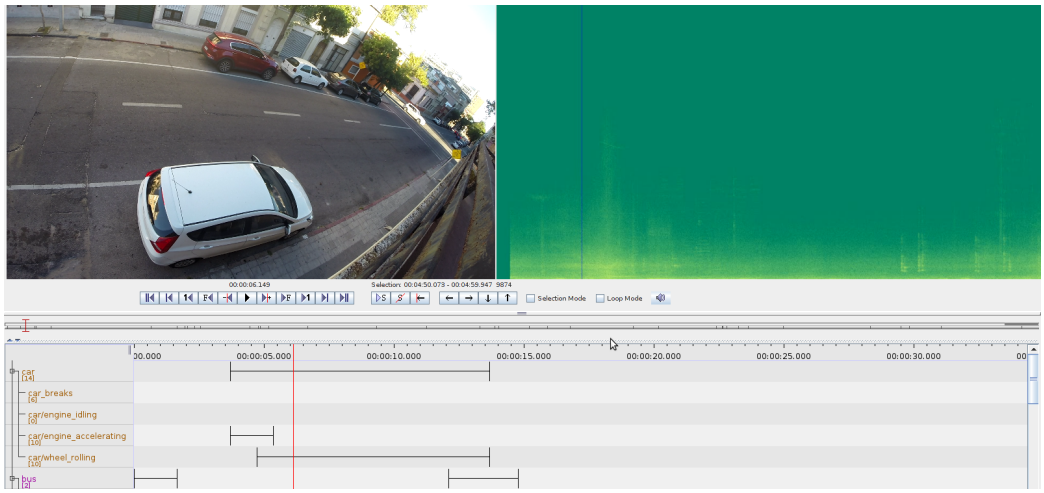
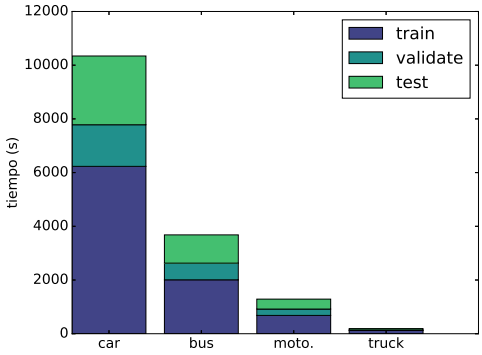


Figura 2.6: Captura de pantalla del software ELAN. Se puede ver la información de video y el espectrograma con un marcador que indica el instante actual. También se pueden ver las anotaciones en el panel inferior.

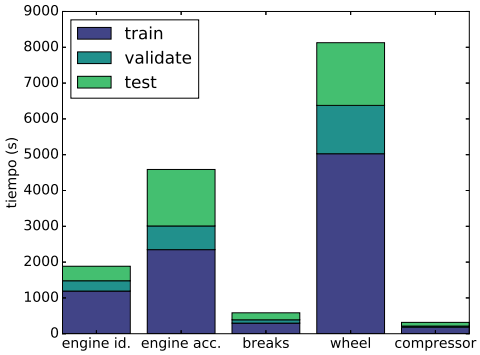
El proceso de etiquetado se realiza en dos etapas. En primer lugar se marcan las etiquetas de las categorías de vehículos (CAR, BUS, etc.). Luego, para cada uno de los segmentos marcados se anotan las etiquetas de las categorías de componentes (ENGINE IDLING, BREAKS, etc.) para formar el nivel subordinado. En la Figura 2.7 se indican las duraciones totales de eventos para los tres tipos de categorías. Notar las clases están muy desbalanceadas, especialmente en el nivel subordinado de categorías. Además, en la Tabla 2.1 se pueden comparar las principales características de las bases presentadas en este capítulo.

Base de datos	Número de clases	Duración total (h)	Anotación
UrbanSound8k	10	8.75	archivos
URBAN-SED	10	30	eventos
TUT	7	1	eventos
SonidosMVD	21	3.8	eventos

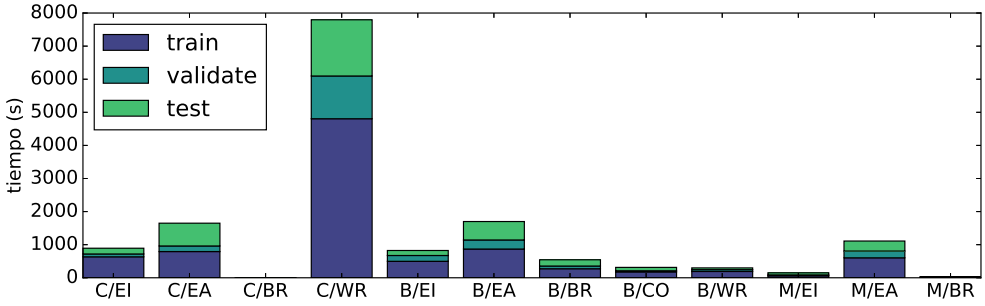
Tabla 2.1: Comparación de principales características de las bases de datos presentadas.



(a) Categorías de vehículos.



(b) Categorías de componentes.



(c) Categorías del nivel subordinado.

Figura 2.7: Duración total de eventos de las categorías de (a) vehículos, (b) componentes y (c) algunas del segundo nivel en los conjuntos de *train*, *validate* y *test* para la base SonidosMVD.

Capítulo 3

Evaluación y medidas de desempeño

> *"Sólo podemos, pues, salir al paso de la injusticia o vaciedad de nuestras aserciones exponiendo el modelo como lo que es, como objeto de comparación –como, por así decirlo, una regla de medir; y no como prejuicio al que la realidad tiene que corresponder."*
>
> – Ludwig Wittgenstein, Investigaciones Filosóficas

En este capítulo se presenta la metodología de evaluación de los sistemas de detección de eventos sonoros utilizada en este trabajo. Además, se definen las medidas de desempeño seleccionadas para evaluar el resultado de los modelos.

3.1. Metodología de evaluación

En la Figura 3.1 se muestra el diagrama general de un sistema de detección de eventos sonoros, donde se puede ver que comúnmente para cada señal de audio se calculan características de tiempo corto (ver Capítulo 4), y se asocian a las anotaciones de referencia (*ground-truth*) con la misma resolución temporal. Con este par característica–etiquetas se entrena el sistema de detección de eventos. Luego, se utilizan nuevos registros de audio como entrada para clasificar eventos y la salida es una matriz de activaciones, que finalmente es transformada en una lista de eventos que incluye el tiempo de inicio, tiempo de fin y la etiqueta correspondiente de cada evento.

Es necesario definir una metodología que permita evaluar el desempeño de estos sistemas de forma justa y que posibilite la comparación de algoritmos [32]. Muchas veces esta metodología es propuesta por los autores que publican las bases de datos. Una metodología correcta conduce a una evaluación estadísticamente válida y evita el sobre-entrenamiento de los sistemas, especialmente cuando los datos están fuertemente desbalanceados [33].

Lo primero que se tiene en cuenta en la evaluación de algoritmos de clasificación es separar los datos en dos conjuntos: uno para entrenar el sistema (*train*) y otro para evaluarlo (*test*). El objetivo de esto es que la evaluación del sistema no se realice sobre los datos que se utilizaron en el entrenamiento, sino que se intente

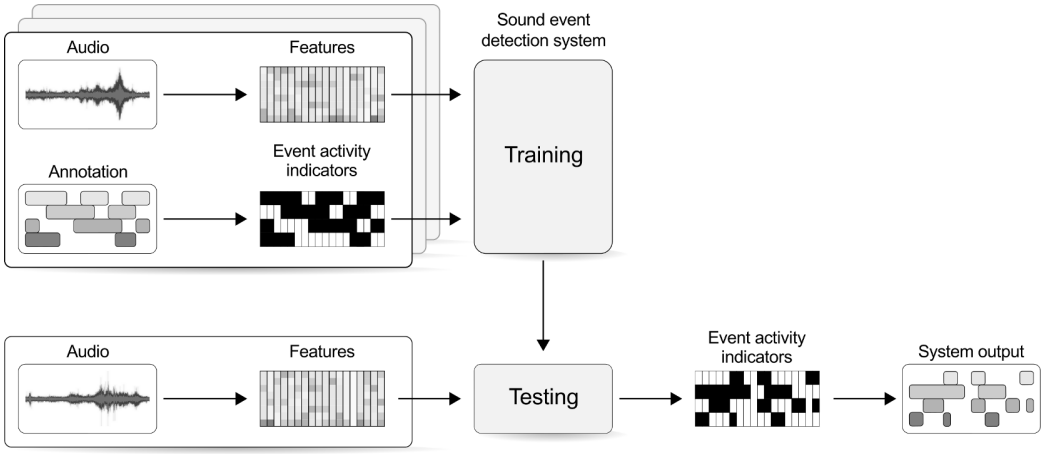


Figura 3.1: Diagrama general del entrenamiento y prueba de un sistema de detección de eventos sonoros extraído de Mesaros et. al. [23].

conocer –o por lo menos estimar– el desempeño en datos no conocidos. De esta forma se busca estimar la capacidad de generalización del sistema. Se busca que los datos de ambos conjuntos sean diferentes en algún sentido, por ejemplo, que sean grabaciones en locaciones diferentes o en horarios diferentes. Luego, el conjunto de *train* se puede dividir en subconjuntos disjuntos (*folds*) y aplicar el método de validación cruzada (*cross validation*), para elegir parámetros del sistema y definir un punto de trabajo. Este método consiste en realizar un experimento para clasificar los datos en cada *fold*, mientras que se entrena el algoritmo con el resto de los *folds* [34].

Esta separación en *folds* también puede estar definida por las características constitutivas de los datos, por ejemplo, cada *fold* puede estar integrado por registros de locaciones diferentes. Cuando no se tienen muchos datos o están muy desbalanceados, la evaluación del sistema para la elección de parámetros muchas veces se realiza sobre un conjunto de validación (*validate*).

Otro punto clave en la evaluación de los sistemas, es la elección de las medidas que se utilizan para estimar su desempeño y posibilitan la comparación de resultados. A continuación se define y se analiza un conjunto de medidas de desempeño que han sido utilizadas en este tipo de problemas.

3.2. Medidas de desempeño

Las medidas propuestas para estimar el desempeño de los sistemas de detección de eventos sonoros son: tasa de error (ER) y F–score ($F1$), sobre una grilla de tiempo fijo [23, 26, 35, 36]. Los eventos sonoros detectados se comparan con el *ground-truth* en segmentos de un segundo. La Figura 3.2 es un diagrama del procedimiento que se aplica para calcular estas medidas. De esta forma, para cada segmento se pueden considerar tres casos:

- Correcto: un evento es considerado correctamente detectado si está presente

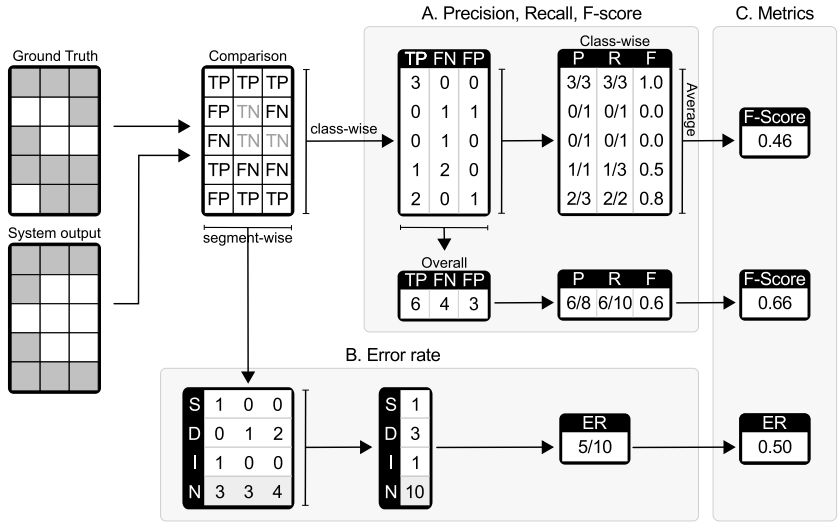


Figura 3.2: Diagrama del procedimiento de cálculo de las medidas de desempeño por segmentos de Mesaros et. al. [23].

en el *ground-truth* y activo en la salida del sistema en dicho segmento.

- Falso Positivo (FP): si la salida del sistema es positiva para un evento que no está presente en el *ground-truth*..
- Falso Negativo (FN): si la salida del sistema es negativa para un evento que sí se encuentra presente en el *ground-truth*.

Basados en las medidas de FP y FN se pueden calcular *precision* y *recall*:

$$P = \frac{TP}{TP + FP},\ R = \frac{TP}{TP + FN}. \tag{3.1}$$

Luego, se calcula el valor de F–score como la media armónica de P y R:

$$F1 = \frac{2PR}{P + R}. \tag{3.2}$$

Como se ve en el diagrama de la Figura 3.2, el valor de $F1$ se puede calcular de forma global, sobre todos los segmentos y clases en simultáneo; o también se puede calcular por clases para luego promediar los valores. Se denomina $\bar{F}1$ a este promedio por clases.

La tasa de error (ER) se calcula en términos de inserciones (I), borrados (D[1]) y sustituciones (S). Para un segmento k, su número está dada por:

$$S(k) = \text{mín}\{FN(k), FP(k)\} \tag{3.3}$$
$$D(k) = \text{máx}\{0, FN(k) - FP(k)\} \tag{3.4}$$
$$I(k) = \text{máx}\{0, FP(k) - FN(k)\}. \tag{3.5}$$

[1]Del inglés *deletions*.

Una sustitución está definida como el caso en que el sistema detecta un evento en un segmento, pero con la etiqueta equivocada. Esto es equivalente a contener un FP y FN en el mismo segmento. Después de contar las sustituciones por segmento, el resto de FP en la salida del sistema son contados como inserciones y el resto de FN como borrados. Luego, la medida de error se calcula integrando las medidas anteriores en todos los K segmentos:

$$ER = \frac{\sum_{k=1}^{K} S(k) + \sum_{k=1}^{K} D(k) + \sum_{k=1}^{K} I(k)}{\sum_{k=1}^{K} N(k)}, \tag{3.6}$$

donde $N(k)$ es el número de clases activas en el *ground-truth* en el segmento k [22, 23].

La elección de ER como medida de desempeño se debe a su utilización en otros problemas como reconocimiento de habla [23], pero tiene algunos problemas. Cuando la salida del sistema es nula –no se detecta nada– el valor de ER es 1 ya que se comenten N errores de borrado. Por otro lado, si el sistema comete más errores que segmentos activos en el *ground-truth*, el valor de ER es mayor a 1. Esto lleva a que sea una medida difícil de interpretar. Para entender mejor esto, se puede ver que los valores de S, D e I se pueden escribir como expresiones algebraicas de la siguiente forma:

$$S = \frac{FN + FP - |FN - FP|}{2} \tag{3.7}$$

$$D = \frac{FN - FP + |FN - FP|}{2} \tag{3.8}$$

$$I = \frac{FP - FN + |FN - FP|}{2}. \tag{3.9}$$

Por lo que, la suma de errores por segmento puede escribirse como:

$$S + D + I = \frac{FP + FN + |FN - FP|}{2}, \tag{3.10}$$

y esto no es otra cosa que el máximo entre FP y FN, por lo que el valor de ER resulta de calcular:

$$ER = \frac{\sum_{k=1}^{K} \text{máx}\{FP(k), FN(k)\}}{\sum_{k=1}^{K} N(k)}. \tag{3.11}$$

Entonces, para cada segmento, simplemente se calcula el máximo entre falsos positivos y falsos negativos y luego se integra por el total de segmentos. Por lo tanto, en esta medida no se tiene en cuenta la relación entre los falsos positivos y falsos negativos, por lo que es difícil seleccionar un punto de trabajo. Si se minimiza ER se puede considerar que se cometen menos errores, pero sin saber de qué tipo.

Otra forma de plantear la evaluación de errores es con la curva de compromiso entre detección y error (*detection error trade–off*, DET) [37], tasa de falsos positivos (FPR) vs. tasa de falsos negativos (FNR), donde dichos valores están dados por:

$$FPR = \frac{\sum_{k=1}^{K} FP(k)}{\sum_{k=1}^{K} N(k)}, \tag{3.12}$$

$$FNR = \frac{\sum_{k=1}^{K} FN(k)}{\sum_{k=1}^{K} N(k)}. \tag{3.13}$$

A partir de estas curvas, es posible calcular la tasa de igual error (*equal error rate*, EER) que se define como el punto en el que FPR es igual a FNR. Al igual que las otras medidas, el valor de EER se puede calcular de forma global o promediando entre el resultado de todas las clases.

3.2.1. Problemas debidos al desequilibrio de clases

Las medidas de desempeño pueden ser muy dependientes del desequilibrio de clases en el conjunto donde se evalúen [38]. Para ilustrar estos efectos en las medidas de ER y $F1$ se presenta el siguiente experimento. Se generan datos aleatorios en una grilla de tiempo fija como la definida anteriormente. Se genera una matriz, GT, de 2000 filas (datos) y 10 columnas (clases) que representa el *ground-truth* de los datos. Cada columna se genera con un proceso estocástico con distribución de Bernoulli con probabilidad diferente para representar el desequilibrio de clases:

$$GT_{i,c} \sim Be(p_c). \tag{3.14}$$

Las probabilidades de cada columna, p_c, siguen una ley logarítmica. Se hacen experimentos con diferentes valores de p_c para simular diferentes niveles de equilibrio. Para estimar qué tan desbalanceadas están las clases en los datos generados artificialmente en cada experimento, se utiliza el coeficiente gini[2].

Experimento 1

En primer lugar, se simula un sistema que detecta perfectamente la clase mayoritaria y no detecta nunca el resto de las clases. Se miden los valores de ER y $F1$ globales y promedios para cada desequilibrio y se grafican en la Figura 3.3 en función del coeficiente gini. Se puede ver que lo valores globales son muy dependientes del desequilibrio, aún más en el caso de $F1$; si el desequilibrio es muy alto, el valor de $F1$ supera el 55 %. Por otro lado, si bien las medidas de ER y F1 promedio no dependen del desequilibrio de clase, en el siguiente experimento se pone en evidencia su dificultad para evaluar desempeño.

Experimento 2

En este otro experimento, se simula un sistema cuya salida es aleatoria para todas clases, por lo tanto se espera que el desempeño del sistema para todas los niveles de desequilibrio sea bajo. En la Figura 3.4 se puede ver que los resultados de $F1$ y ER promedio son peores que los globales para todos los niveles de desequilibrio, aunque todavía no es claro que sea deseable que un sistema de 10 clases con salida aleatoria obtenga esos resultados.

[2]Se utiliza la curva de Lorentz para medir la diferencia entre la distribución de una variable (en este caso la duración total de los eventos de cada clase) y una distribución perfectamente uniforme. El coeficiente varia entre 0 (perfecto equilibrio) y 1 (perfecto desequilibrio, los datos pertenecen todos a la misma clase).

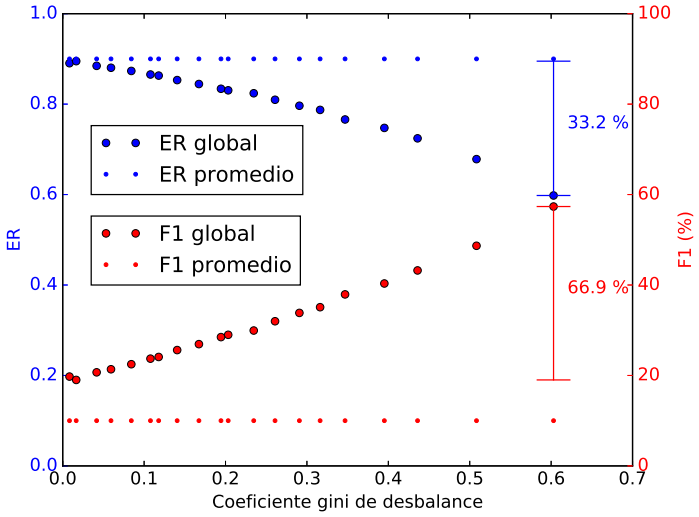


Figura 3.3: Resultados de ER y $F1$ globales y promedios del experimento 1. Sistema simulado que detecta sólo la clase mayoritaria en datos con diferentes niveles de desequilibrio.

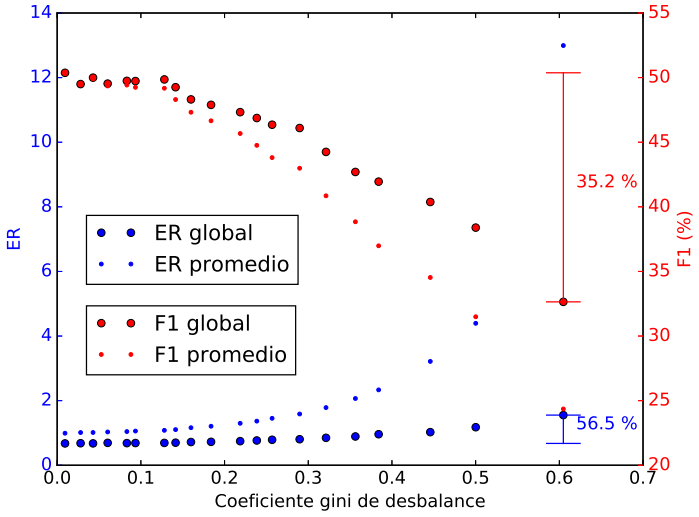


Figura 3.4: Resultados de ER y $F1$ globales y promedios de un sistema simulado que genera salidas aleatorias para todas las clases en datos con diferentes niveles de desequilibrio.

Además de lo planteado anteriormente, en los resultados de las competencias DCASE 2016 y 2017, se puede ver que los algoritmos que obtienen los 'mejores' resultados globales, en realidad solo detectan muy bien la clase mayoritaria [39,40].

3.2.2. Propuesta de nuevas medidas de desempeño

Se propone una nueva forma de integrar los valores de ER y $F1$ por clase, de manera de mejorar la evaluación de este tipo de sistemas con salida multi–clase y

con datos muy desbalanceados. Lo que se busca es aumentar el peso de la detección de las clases minoritarias en el desempeño del sistema [41].

El promedio por clase de las medidas de desempeño que se hace anteriormente es simplemente:

$$\bar{M} = \frac{1}{C}\sum_{c=1}^{C} M_c, \tag{3.15}$$

donde C es el número de clases y M_c es la medida de desempeño ($F1$ o ER) para la clase c. Ahora se propone realizar un promedio ponderado por alguna medida del desequilibrio de las clases:

$$\hat{M} = \sum_{c=1}^{C} w_c M_c, \tag{3.16}$$

donde w_c es el peso fijado para cada clase. Para darle más importancia a las clases minoritarias, simplemente se fijan los pesos de la siguiente forma:

$$w_c = \frac{1/N_c}{\sum_{j=1}^{C} 1/N_j}, \tag{3.17}$$

donde N_c es el número de segmentos activos para la clase c. Notar que w_c aumenta cuando N_c disminuye, como es esperado. La suma en el denominador simplemente asegura que $\sum_c w_c = 1$.

Para evaluar la capacidad de la nuevas medidas de desempeño y compararlas con las anteriores, se plantea un nuevo experimento simulado. Esta vez se utilizan las referencias (*ground-truth*) del conjunto de validación de la base de datos SonidosMVD que fue grabada en condiciones reales y por lo tanto tiene un gran desequilibrio de clases (ver Figura 2.7). Los experimentos se realizan con tres clases del nivel básico de vehículos (Car, Bus y Motorcycle). En la Tabla 3.1 se comparan los resultados de cuatro tipos de sistemas: 1) salida completamente aleatoria; 2) sistema que detecta sólo una de las clases; 3) sistema que detecta dos clases correctamente; 4) sistema que confunde una clase con otra.

Lo primero que llama la atención para las medidas globales es que el resultado de la salida aleatoria es muy alto. Además, al detectar solamente la clase mayoritaria (Car) se obtiene un muy buen desempeño. En el caso de los promedios por clase, el sistema con salida aleatoria obtiene un resultado pobre (más coherente), pero luego los valores para todas las clases son obviamente los mismos. Luego, para las sumas ponderadas se puede ver que los resultados de error son similares a los globales para la detección de clases, pero es mucho mayor para la salida aleatoria lo cual es muy deseable. Por último notar que los valores de $\hat{F}1$ son más estrictos que sus pares globales y que representan con mayor fidelidad el desequilibrio, ya que se premia más la detección de las clases minoritarias. Esto último se puede confirmar al ver los resultados de los sistemas que confunden una clase con otra.

También se puede hacer un estudio similar sobre la medida de *equal error rate* (EER), pero en este caso es necesario que la salida simulada del sistema no sea binaria y que la salida se pase por un umbral variable de forma de generar la curva

	Error Rate			F–score (%)		
Experimento	ER	$\bar{ER}$	$\hat{ER}$	$F1$	$\bar{F1}$	$\hat{F1}$
Salida aleatoria	1.00	2.06	2.85	43.9	39.2	29.2
Solo Car	0.36	0.66	0.89	77.9	33.3	11.0
Solo Bus	0.75	0.66	0.72	39.7	33.3	28.4
Solo Moto.	0.88	0.66	0.39	20.8	33.3	60.6
Car y Bus	0.12	0.33	0.61	93.8	66.6	39.4
Car y Moto.	0.25	0.33	0.28	85.9	66.6	71.6
Bus y Moto.	0.64	0.33	0.11	53.3	66.6	89.0
Car por Bus	0.75	1.52	1.45	26.2	18.6	11.0
Car por Moto.	0.86	2.43	3.61	15.5	12.0	12.9
Moto. por Bus	0.94	1.61	2.04	8.89	11.1	14.8

Tabla 3.1: Resultados de ER y $F1$ globales, promedios por clase ($\bar{ER}$ y $\bar{F1}$) y sumas ponderadas ($\hat{ER}$ y $\hat{F1}$) en el conjunto de validación de SonidosMVD para el experimento simulado de detectar sólo una clase, detectar dos clases y confundir dos clases.

DET. Se simula un sistema cuya salida es completamente aleatoria para toda las clases y otros cuya salida tiene más probabilidad de ser correcta para una de las clases. En la Tabla 3.2 se comparan los valores de la medida global (EER), del promedio por clase ($E\bar{E}R$) y del promedio ponderado ($E\hat{E}R$). En primer lugar es interesante notar que la medida global de EER es más robusta que en las medidas anteriores para datos mas desbalanceados. El problema que tiene la medida global es que el resultado del sistema aleatorio no es sensiblemente mayor que para la detección de las clases. Lo mismo pasa con $E\bar{E}R$ y $E\hat{E}R$ donde el desempeño del sistema con salida aleatoria y del sistema con detección predominante de Car son similares.

Experimento	$EER(\%)$	$E\bar{E}R(\%)$	$E\hat{E}R(\%)$
Salida aleatoria	60.6	60.2	73.5
Predominante Car	42.4	52.4	72.6
Predominante Bus	43.4	35.7	51.6
Predominante Moto.	43.4	33.4	23.8

Tabla 3.2: Valores de EER, promedio por clase $E\bar{E}R$ y suma ponderada $E\hat{E}R$ para un sistemas con salida aleatoria, y salidas con detección predominante de cada una de las clases.

3.3. Discusión

En este capítulo se muestra la metodología de evaluación de los sistemas diseñados en esta tesis con especial énfasis en la elección de la medida de desempeño. Se puede concluir que, dadas las características de los problemas atacados, la elección de la medida de desempeño es un punto especialmente desafiante.

Difícilmente sistemas tan complejos como los estudiados en este trabajo puedan evaluarse con una única medida, pero por la necesidad de presentar y comparar los resultados de forma simple, se elige muchas veces ese camino. La elección de una forma de evaluación es necesariamente subjetiva y condiciona todo el trabajo que se realiza sobre el problema.

En los experimentos presentados anteriormente, se muestra que las medidas más utilizadas para evaluar los algoritmos del estado del arte, $F1$ y ER globales, no son muy confiables cuando se tienen conjuntos muy desbalanceados. Se propone una nueva forma de ponderar las medidas por cada clase con el objetivo de obtener una evaluación más exigente. La elección de este tipo de promedio busca penalizar a los algoritmos que por el desequilibrio de clases, tienden a detectar solo las clases mayoritarias. Como contrapartida, si la detección de las clases minoritarias no aporta mucho a lo que se busca con el sistema, estas medidas ponderadas pueden agregar ruido innecesario.

En este trabajo se intenta evitar el uso de una sola (o dos) medidas para evaluar sistema complejos. En el estudio del problema de clasificación de registros sonoros (Apéndice A) se acompañan la medidas por matrices de confusión que condensan mucha información sobre los aciertos y errores de los algoritmos. Para el problema de detección de evento sonoros (Capítulos 5 y 6), se utilizan las medidas presentadas en la sección anterior acompañadas de las curvas DET que también condensan mucha información sobre las tasas de error y ayudan a elegir un punto de trabajo.

Capítulo 4

Extracción de características

> *"... dividir cada una de las dificultades que examinase en tantas partes como fuera posible y como se requiriese para su mejor resolución... [Luego] conducir ordenadamente mi pensamiento comenzando por los más simples y fáciles de conocer para ascender poco a poco, como por grados, hasta el conocimiento de los más complejos, suponiendo, incluso, un orden entre los que no se preceden naturalmente."*
>
> – René Descartes, Discurso del método

En este capítulo se presentan las principales características utilizadas para representar las señales de audio en los sistemas de clasificación y detección expuestos en los capítulos siguientes. Se muestra que dichas características son muy utilizadas en los modelos del estado del arte, aunque recientemente se han diseñando redes neuronales profundas cuya entrada es la señal de audio sin procesar como se ve en la Sección 6.6.

4.1. Análisis de tiempo corto

4.1.1. Espectrograma

Usualmente las señales de audio se transforman a un dominio frecuencial con el objetivo de obtener un mapa de representación tiempo–frecuencia. Para pasar a un domino de frecuencias en una escala de representación lineal se utiliza la transformada discreta de Fourier (*Discrete Fourier Transform*, DFT). Este paso al dominio de las frecuencias se realiza para cada fragmento de corta duración (*frame*) de la señal de audio de forma de obtener la transformada de Fourier de tiempo corto (*Short Time Fourier Transform*, STFT). Si $x[n]$ es la señal en el dominio del tiempo, la STFT para una componente de frecuencia k y un *frame* i, se calcula como:

$$X[i,k] = \sum_{m=0}^{N} w[m]x[iN_h + m] \exp\left(\frac{-j2\pi km}{N}\right) \tag{4.1}$$

donde $w[m]$ es la ventana de análisis, N es el tamaño de la ventana, N_h es el número de muestras de salto (*hop*) entre cada $frame$. La STFT se calcula eficientemente utilizando la transformada rápida de Fourier (*Fast Fourier Transform*, FFT). Se denomina espectrograma a la representación tiempo–frecuencia que surge de calcular el módulo de la STFT en cada punto. Aunque el espectrograma es utilizado en sistemas de detección de eventos sonoros, por ejemplo como entrada de redes neuronales recurrentes [42] o convolucionales [43], no es el tipo de características más utilizadas, ya que se obtienen mejores resultados con otras representaciones de la señal de audio basadas en modelos de percepción auditiva.

4.1.2. Energía en bandas de frecuencia en escala mel

Una forma de integrar la información del espectrograma basándose en un modelo de percepción auditiva es reducir el número de componentes de frecuencias aplicando un banco de filtros en las frecuencias mel (representado en la Figura 4.1). Este banco está formado por filtros triangulares, cuyo ancho de banda es constante hasta 1000 Hz y constante en la escala mel para frecuencias superiores. Esta escala está definida por:

$$\text{mel}(f) = 1127.01048 \log\left(1 + \frac{f}{700}\right). \tag{4.2}$$

El ancho de banda y la distribución logarítmica de este banco de filtros busca simular la resolución en frecuencia del sistema auditivo humano.

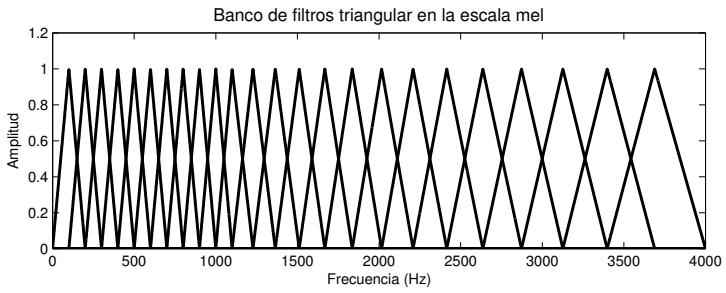


Figura 4.1: Banco de filtros en la escala mel

La energía en las bandas mel se puede calcular en el dominio de la frecuencia de la siguiente forma:

$$E[i,l] = \sum_k |H_l[k]X[i,k]|^2,\ l = 0, \ldots, L-1, \tag{4.3}$$

donde $H_l[k]$ es la transferencia del l–ésimo filtro y L es el número de filtros utilizado. Como este cálculo se hace para cada uno de los *frames* de la señal, se obtiene un mapa tiempo–frecuencia que representa la distribución de la energía en las bandas en la escala mel para cada *frame* de la señal.

La energía en las bandas mel son utilizadas para atacar los problemas definidos en este trabajo, junto con redes recurrentes [44–47], redes convolucionales [19, 48] y redes convolucionales-recurrentes [43, 49].

4.1.3. Mel–Frequency Cepstral Coefficients (MFCC)

Los coeficientes cepstrales en las frecuencias mel (MFCC) fueron originalmente concebidos para describir el contenido espectral de señales de voz, ya que sirven para descomponer ese tipo de señales en una excitación (portadora) y su modulación (respuesta de un filtro). En el caso de los modelos de generación de voz, la excitación busca representar la señal generada en la glotis y el filtro modela los efectos del tracto vocal. Con los coeficientes MFCC se busca encontrar la respuesta en frecuencia de ese filtro, por lo que sirven para identificar hablantes o reconocer el habla. Los MFCC también han sido utilizados con éxito en diversos problemas vinculados a la música, como identificación de cantantes [50] o reconocimiento de instrumentos musicales [51]. Este tipo de características también ha sido aplicado a la clasificación del entorno sonoro junto con modelos gaussianos [52], árboles de decisión [16, 18, 53] y redes neuronales profundas [54–57].

Para calcular los coeficientes cepstrales se calcula la Transformada discreta del coseno (DCT) del logaritmo de la energía en las bandas de frecuencia mel:

$$\mathrm{MFCC}[i,m] = \sum_{l=0}^{L-1} \log(E[i,l]) \cos\left(\frac{2\pi}{L} lm\right),\ m = 0, \ldots, M-1, \qquad (4.4)$$

donde M es el número de coeficientes cepstrales [58].

En la Figura 4.2 se ilustran todos los pasos para calcular los MFCC. De la forma de onda de la señal, se extrae el espectrograma, la energía en las bandas mel y por último los coeficientes cepstrales.

4.2. Integración temporal

Las representaciones presentadas en las secciones anteriores son de tiempo corto, describen la señal para cada *frame*. En algunos casos es de interés integrar la información de esas características de tiempo corto en ventanas temporales mayores. Esto es de especial interés para el problema de clasificación del Apéndice A, ya que se busca clasificar cada registro sonoro en su totalidad y no a nivel de *frame*. Para realizar esta integración se suelen calcular diferentes estadísticas como la media y la varianza. Además se pueden calcular estadísticas sobre las derivadas de cada característica. A modo de ejemplo, para obtener la media de la energía en las bandas mel de una señal con N *frames*, se calcula:

$$\bar{E}[l] = \frac{1}{N} \sum_{i=0}^{N-1} E[i,l]. \qquad (4.5)$$

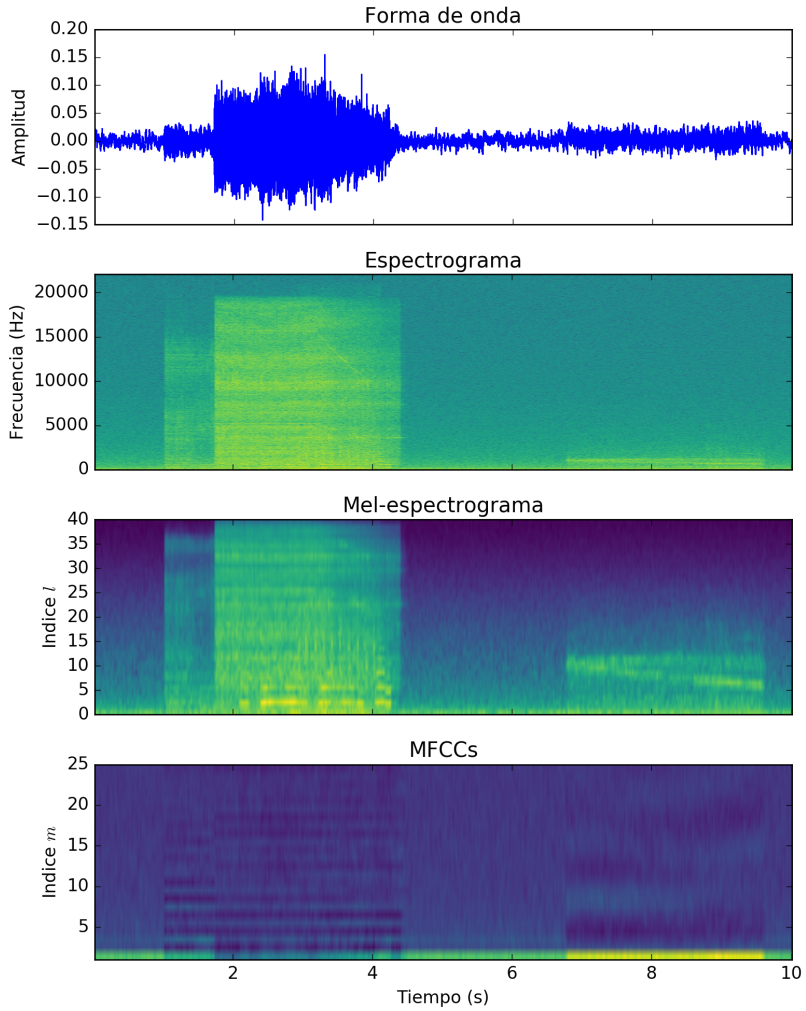


Figura 4.2: A partir de la señal de audio se extrae el espectrograma, la energía en las bandas mel y los MFCC. En este caso $L = 40$ y $M = 25$.

En el problema de detección de eventos sonoros, donde se busca clasificar todas las fuentes presentes, incluyendo los tiempos de comienzo y final, puede ser necesario trabajar a nivel de *frame*. En estos casos, la integración temporal se puede realizar dentro de los modelos de clasificación. Por ejemplo, cuando se trabaja con técnicas de aprendizaje profundo (Capítulo 6), la propia estructura de las redes neuronales permite agrupar las características en las capas de *pooling* o integrarlas con operaciones recursivas (ver Sección 6.4).

4.3. Normalización de energía

En general, la energía en las bandas mel, $E[i,l]$, se convierte a decibeles de la siguiente forma:

$$E_{dB}[i,l] = 10\log_{10}\left(E[i,l]\right). \tag{4.6}$$

Este tipo de normalización es muy utilizado en el problema de detección de eventos sonoros [19, 49]. Recientemente se propuso otra forma de normalización denominada *Per-Channel Energy Normalization* (PCEN), diseñada para aumentar la robustez a las variaciones de intensidad en sistemas de detección de habla [59]. Con PCEN se remplaza la función logarítmica estática por una compresión de rango dinámico (DRC) y un control adaptativo de ganancia (AGC) con integración temporal. Esta integración se realiza con un filtro pasabajos ϕ_T a una escala T, por lo que se tiene la siguiente expresión:

$$E_{PCEN}\left[i,l\right] = \left(\frac{E\left[i,l\right]}{\left(\epsilon + M\left[i,l\right]\right)^{\alpha}} + \delta\right)^{r} - \delta^{r}, \tag{4.7}$$

donde $M\left[i,l\right] = \left(E^{t} * \phi_T\right)\left[i,l\right]$ y α, ϵ, r y δ son constantes positivas [60]. En la Figura 4.3 se puede ver la diferencia entre el escalado logarítmico la normalización utilizando PCEN.

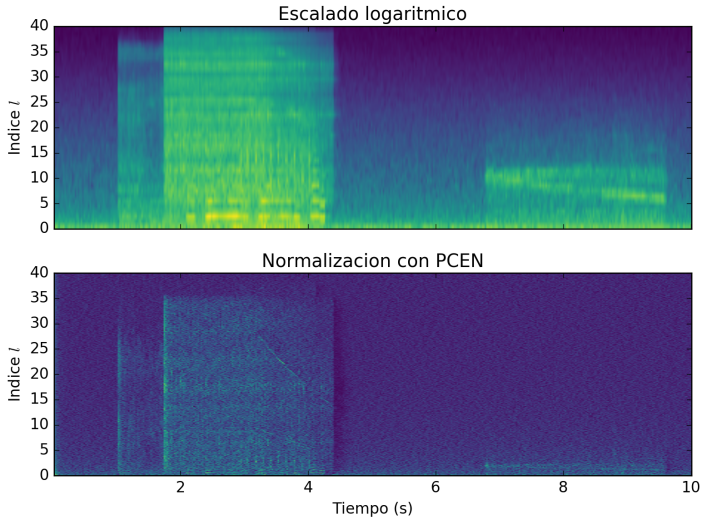


Figura 4.3: Comparación de normalización de energía con la función logaritmo (arriba) y utilizando PCEN (abajo). La función PCEN se calcula utilizando *librosa* [60, 61] con parámetros $\alpha = 0.8$, $\delta = 10$, $r = 0.25$, $T = 0.06$, $\epsilon = 10^{-6}$

Se han propuesto valores para los parámetros de PCEN en función de estudios asintóticos [60] pero es interesante ver que la función es diferenciable y por lo tanto dichos parámetros pueden ser aprendidos en sistemas con redes neuronales [59]. En el Capítulo 6 se hace un estudio en este sentido.

4.4. Extracción de características usando redes neuronales

En los últimos años las técnicas de aprendizaje profundo han sido parte del estado del arte en diversas áreas. En el Capítulo 6 se presentan algunas de estas técnicas y se estudia su impacto en el problema atacado en esta tesis. Gran parte del éxito de estas técnicas se debe a su aplicación al problema de clasificación de imágenes y la disponibilidad de grandes bases de datos como ImageNet [62]. Se entiende que las primeras capas de estas redes neuronales *aprenden* las características más relevantes para la discriminación de las clases. Por ejemplo, en el caso de la clasificación de imágenes, la primera capa de convolución aprende filtros muy variados en cuanto a orientación, frecuencia y selectividad, además de varias regiones (*blobs*) coloreadas [62]. Por lo tanto, estos filtros extraen características de borde, forma, textura y color que resultan útiles para la tarea de clasificación.

Por otro lado, en el dominio del audio, se ha mostrado que en las primeras capas de este tipo de redes, se extraen características de bajo nivel similares al espectrograma [63], la energía en las bandas en escala mel [64] o *wavelets* [65]. En la Figura 4.4 se pueden ver los filtros que se aprenden en la primera capa en [65]. En el Capítulo 6 se profundiza sobre el aprendizaje de características en redes punta a punta (*end–to–end*).

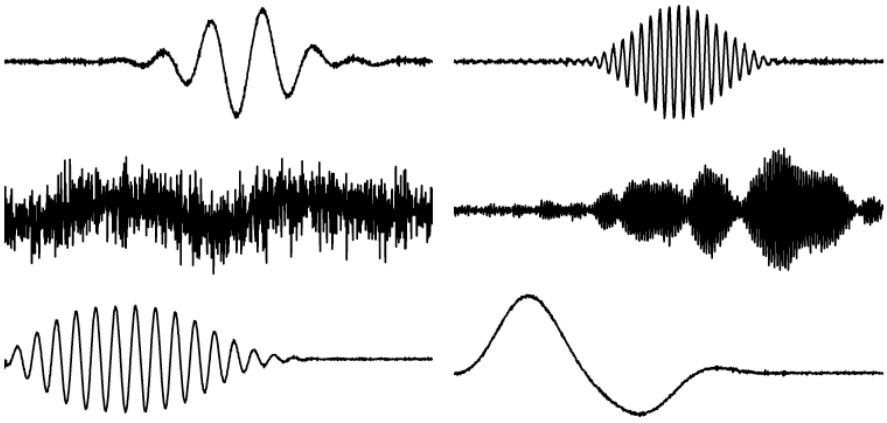


Figura 4.4: Algunos de los filtros que se aprenden en la primera capa de [65].

Capítulo 5

Detección basada en aprendizaje poco profundo (shallow learning)

En este capítulo se abordan técnicas de aprendizaje poco profundo para la detección de eventos sonoros. Se llama aprendizaje poco profundo al enfoque que involucra el procesamiento de las señales de audio para extraer características y el reconocimiento automático de patrones (SVM, RandomForest, etc.) para clasificar en distintas clases. Se denomina de esta forma para diferenciarlo del aprendizaje profundo con redes neuronales de varias capas que se muestra en el Capítulo 6. Para los experimentos reportados en este capítulo se utilizan las bases de datos TUT y SonidosMVD (ver Capítulo 2).

Para desarrollar y medir el desempeño de los modelos (ver Capítulo 3), se utiliza una plataforma de trabajo desarrollada en Python por los organizadores del desafío DCASE [22]. En esta plataforma se dispone de un sistema de extracción y normalización de características, y un sistema de entrenamiento, clasificación y evaluación de modelos. Para la extracción de características se utiliza la biblioteca *librosa* [61] y para entrenar los modelos se utiliza *scikit-learn* [66].

5.1. Características

En los sistemas presentados en este capítulo, se utilizan 20 coeficientes MFCC que son calculados con la energía en 40 bandas mel (ver Capítulo 4). Estas características son calculadas en *frames* de 40 ms con ventana de análisis de Hamming solapadas un 50 %. Además, para poder describir las variaciones temporales de los coeficientes, se calculan sus derivadas de primer y segundo orden (ΔMFCC, Δ^2MFCC).

El coeficiente MFCC[0] $= \sum_{l=0}^{L-1} \log(E[l])$, es una medida de la potencia de la señal, por lo que en algunos casos, no se utiliza para independizar la clasificación de esa característica. Por lo tanto, en total se utilizan 20 (o 19) MFCC, 20 ΔMFCC y 20 Δ^2MFCC, totalizando 60 (o 59) características.

5.2. Modelos

En esta sección se presentan cuatro modelos de clasificación cuyas entradas son el conjunto de características presentadas anteriormente. En la siguiente sección se comparan los resultados de dichos modelos.

5.2.1. Baseline - GMM

En el desafío DCASE se propone un sistema de base (*baseline*) como referencia de desempeño. En este sistema se entrenan 7 modelos (uno para cada clase) independientes en una lógica de *una clase contra el resto*. Para cada clase, se entrenan dos modelos de mezclas de guassianas (*Gaussian Mixture Model*, GMM), uno para las muestras pertenecientes a la clase y otro para el resto de los datos. De esta forma, para cada clase ω_c se ajustan dos densidades de probabilidad con la siguiente forma:

$$p(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k), \tag{5.1}$$

donde K es el número de componentes gaussianas. Para cada componente k, se tiene la media μ_k, la matriz de covarianza $\mathbf{\Sigma}_k$ y su peso π_k. Se asume que las matrices de covarianza son diagonales. Entonces, sean

$$p_{\omega_c}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k) \text{ para } \mathbf{x} \in \omega_c \tag{5.2}$$
$$p_{\bar{\omega_c}}(\mathbf{x}|\bar{\mu_k}, \bar{\mathbf{\Sigma}_k}) \text{ para } \mathbf{x} \notin \omega_c \tag{5.3}$$

las densidades modeladas para las muestras pertenecientes a la clase ω_c y las no pertenecientes a la clase ω_c respectivamente.

Para clasificar una muestra nueva, se define una función de decisión $g(\mathbf{x})$ como la resta de las log-probabilidades de que la muestra pertenezca a la clase ω_c (*positiva*) y que pertenezca a la clase $\bar{\omega_c}$ (*negativa*). Si el resultado de dicha resta supera cierto umbral γ se dice que la muestra pertenece a la clase ω_c.

5.2.2. GMM con inicialización supervisada

Como primera aproximación al problema, se propone inicializar los parámetros μ_k y π_k de forma supervisada. Con esto se busca que las densidades de probabilidad estimadas modelen mejor el efecto del solapamiento temporal de eventos sonoros. Por lo tanto, se dividen las muestras pertenecientes a cierta clase ω_c en siete conjuntos. El primer conjunto está formado por las muestras que sólo son de dicha clase, los seis restantes incluyen las muestras que son de esa clase y también de otra. Para fijar ideas, sea D_c el conjunto de muestras de la clase ω_c. Debido al solapamiento temporal, una muestra $\mathbf{x}_i$[1] puede pertenecer a más de un conjunto D_c. De esta forma se dividen las muestras de entrenamiento de la clase ω_c en los

[1]En este caso, $\mathbf{x}_i$ es el vector de características correspondientes al *frame* i.

siguientes conjuntos:

$$S_c^1 = \left\{ \mathbf{x} : \mathbf{x} \in D_c - \bigcup_{j \neq i} D_j \right\} \tag{5.4}$$

$$S_c^p = \left\{ \mathbf{x} : \mathbf{x} \in D_c \cap \left(D_p - \bigcup_{j \neq i,p} D_j \right) \right\} \text{ para } p = 2 \dots C, \tag{5.5}$$

$$\tag{5.6}$$

donde C es el número de clases (7 en este caso). Se puede calcular los pesos de cada conjunto S_c^p como la cantidad de elementos sobre la totalidad de muestras:

$$\pi_p = \frac{\#S_c^p}{\sum_{j=1}^{C} \#S_c^j}. \tag{5.7}$$

Luego, para cada conjunto S_c^p se buscan $K\pi_p$ medias utilizando *k-means*. De esta forma, se busca distribuir las K componentes de la mezcla de gaussianas según la cantidad de muestras en cada conjunto. Por lo tanto, con este método se busca que la densidad de probabilidad de las muestras pertenecientes a la clase y la densidad de las no pertenecientes, modelen correctamente los solapamientos entre clases.

5.2.3. Random Forest

Otro modelo que se prueba es Random Forest. En primer lugar se utilizan las mismas características que las secciones anteriores (MFCC y sus derivadas de primer y segundo orden). Se utiliza la implementación disponible en *scikit-learn* con 100 árboles. El resto de los parámetros se mantiene por defecto, por ejemplo, la cantidad máxima de características utilizadas para dividir los nodos es la raíz cuadrada del número de características ($\sqrt{N_{features}}$).

Como se entrenan modelos independientes para cada clase con la lógica de *uno contra el resto*, las muestras correspondientes a la clase *positiva* son muchas menos que las pertenecientes a la clase *negativa*. Por lo tanto, a la hora de dividir los nodos, el algoritmo le da más peso a la clase negativa. Para variar el punto de trabajo y encontrar el óptimo, se varía el peso que se asigna a la clase positiva en cada uno de los modelos , de forma de cambiar las probabilidades *a priori*. Con los parámetros por defecto, ambas clases tienen peso 1; lo que se debe hacer entonces es aumentar el peso de la clase minoritaria hasta el máximo dado por la relación de muestras entre las clases. De esta forma se varía el peso de la clase minoritaria con la siguiente función:

$$W = 1 + (M - 1)\alpha, \tag{5.8}$$

donde M es la relación entre la clase mayoritaria y la minoritaria y α varía entre 0 y 1. De este modo, se busca que el peso que se quiere variar sea independiente de la cantidad de muestras.

Otra opción es estimar las probabilidades *a posteriori* de que las muestras pertenezcan a la clase positiva, ω_i, para obtener una función de decisión:

$$g(\mathbf{x}) = P(\omega_c|\mathbf{x}) \tag{5.9}$$

donde la probabilidad $P(\omega_c|\mathbf{x})$ es la media de la probabilidad $P_t(\omega_c|\mathbf{x})$ de cada árbol t. La probabilidad $P_t(\omega_c|\mathbf{x})$ se calcula como la fracción de muestras que pertenecen a la clase ω_c en el último nodo luego de aplicar las decisiones. En el funcionamiento por defecto, si $g(\mathbf{x})$ es mayor a 0.5, se decide que la muestra pertenece a la clase positiva. Para modificar el punto de trabajo, se puede variar dicho umbral tomando:

$$g(\mathbf{x}) \underset{\omega_c}{\overset{\bar{\omega_c}}{\lessgtr}} \gamma. \tag{5.10}$$

Este método tiene como ventaja que no se debe entrenar un nuevo modelo para cada valor del umbral, ya que solo se utiliza en el momento de clasificar las muestras.

5.2.4. SVM

Se prueba el algoritmo de C-Support Vector Machines en su implementación de *scikit-learn* con un *kernel* RBF. Para ajustar los parámetros C y γ se hace una grilla de búsqueda con el objetivo de minimizar ER. Este procedimiento se realiza con validación cruzada sobre el conjunto de entrenamiento. En primer lugar se busca en una grilla exponencial dada por:

$$C = \{10^{-3}, 10^{-2}, \ldots, 10^{1}\} \tag{5.11}$$
$$\gamma = \{10^{-4}, 10^{-3}, \ldots, 10^{0}\}. \tag{5.12}$$

El mínimo de ER se da para valores de $C = 0.1$ y $\gamma = 0.001$ y corresponde al punto ($ER = 0.78, F1 = 42.67\,\%$). Luego, se busca en un entorno de esos valores en una escala lineal dada por:

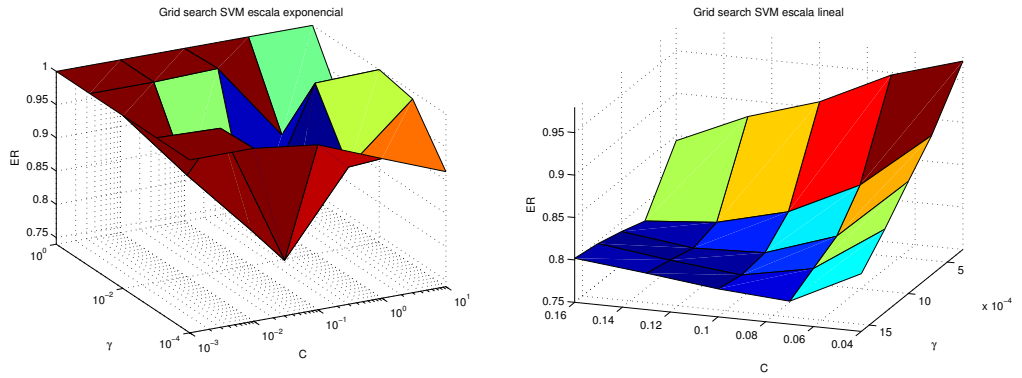
$$C = \{0.03, 0.07, \ldots, 0.16\} \tag{5.13}$$
$$\gamma = \{0.0003, 0.0007, \ldots, 0.0016\}. \tag{5.14}$$

Se encuentra que los parámetros óptimos son $C = 0.13$ y $\gamma = 0.001$ (ver Figura 5.1).

Para este modelo, la función de decisión $g(\mathbf{x})$ está dada por la distancia al hiperplano de separación.

5.3. Resultados

Se entrenan los modelos de la Sección 5.2 con los datos del conjunto de *train* de TUT y se evalúan con todos los datos del conjunto de *test*. Para encontrar las curvas DET se varía el umbral de detección γ y se calculan la cantidad de FP y



(a) Escala exponencial (b) Escala lineal

Figura 5.1: Grillas de búsqueda para SVM.

FN. Los puntos de trabajo se definen en $\gamma = 160$ para ambos modelos GMM y en $\gamma = 0.5$ para SVM y Random Forest.

En primer lugar, se estudia el resultado comparativo del *baseline* con la inicialización supervisada. Los resultados para el experimento de validación cruzada en el conjunto *train* muestran que la inicialización supervisada mejora levemente el desempeño de GMM (ver Tabla 5.1). Mientras que si se analizan los resultados en el conjunto *test*, la conclusión cambia. Para ello, en primer lugar, se pueden ver las curvas DET de cada clase en la Figura 5.2. Se puede apreciar que con la inicialización supervisada se empeora significativamente la detección de WIND BLOWING. En la Figura 5.3 se presentan los resultados de $F1$, ER y EER para cada clase; además se muestran los valores globales, los promedios y las sumas ponderadas. Notar que para las tres medidas, el desempeño de la inicialización supervisada es peor. Esta diferencia entre los resultados en el conjunto de *train* y el de *test* parece indicar que la supervisión genera que el clasificador se sobre–entrene. Esto se puede deber al hecho de que lo que se busca con la inicialización supervisada es que las densidades de probabilidad estimadas describan mejor al conjunto de los datos; y como esta estimación se realiza sobre el conjunto *train*, si las muestras del conjunto de *test* tienen otra distribución, se puede estar sobre–entrenando el modelo.

Sistema	ER	$F1(\%)$
GMM - baseline	0.85	39.0
GMM - supervisado	0.82	40.7

Tabla 5.1: Resultados globales de ER y $F1$ para los clasificadores GMM supervisado y no supervisado en el experimento de validación cruzada con el conjunto *train*.

Ahora se comparan los resultados de los clasificadores SVM y Random Forest contra el *baseline*. En el caso de Random Forest, se entrenan dos opciones, una cuyas características incluyen el coeficiente MFCC[0] y otra que no. En la Tabla 5.2

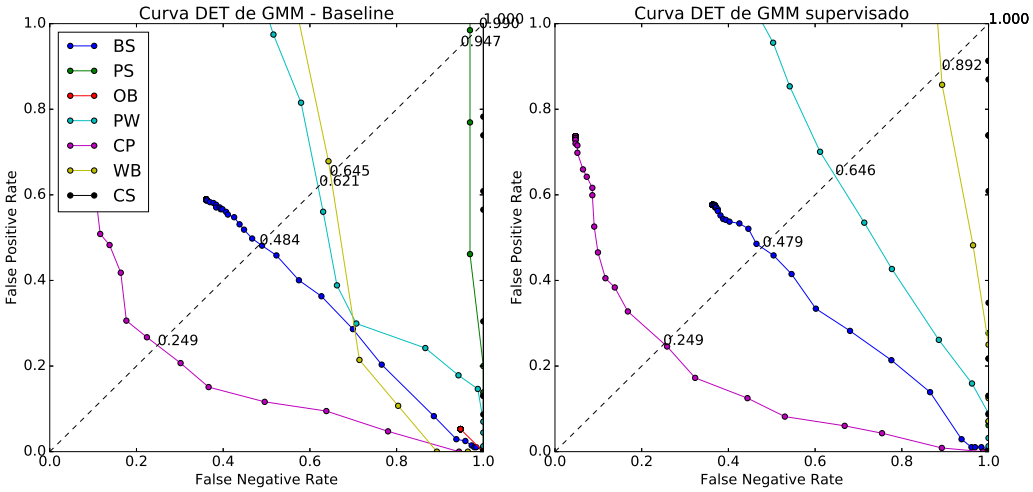


Figura 5.2: Curvas DET de los clasificadores *baseline* y GMM con inicialización supervisada para cada clase.

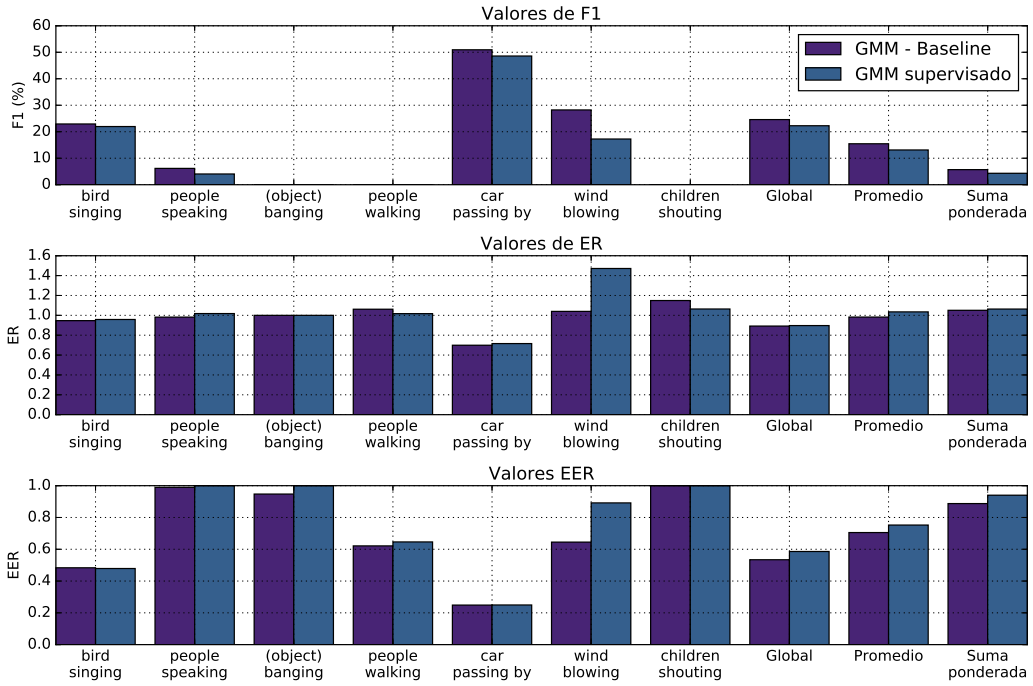


Figura 5.3: Resultados de $F1$, ER y EER para cada clase, globales, promedios y promedios ponderados para los clasificadores del *baseline* y GMM con inicialización supervisada.

se pueden ver los resultados de ER y $F1$ para la validación cruzada en el conjunto de *train* y en la Figura 5.4 se encuentran los resultados detallados para el conjunto de *test*. Notar que en los puntos de trabajo elegidos, todos los clasificadores tienden

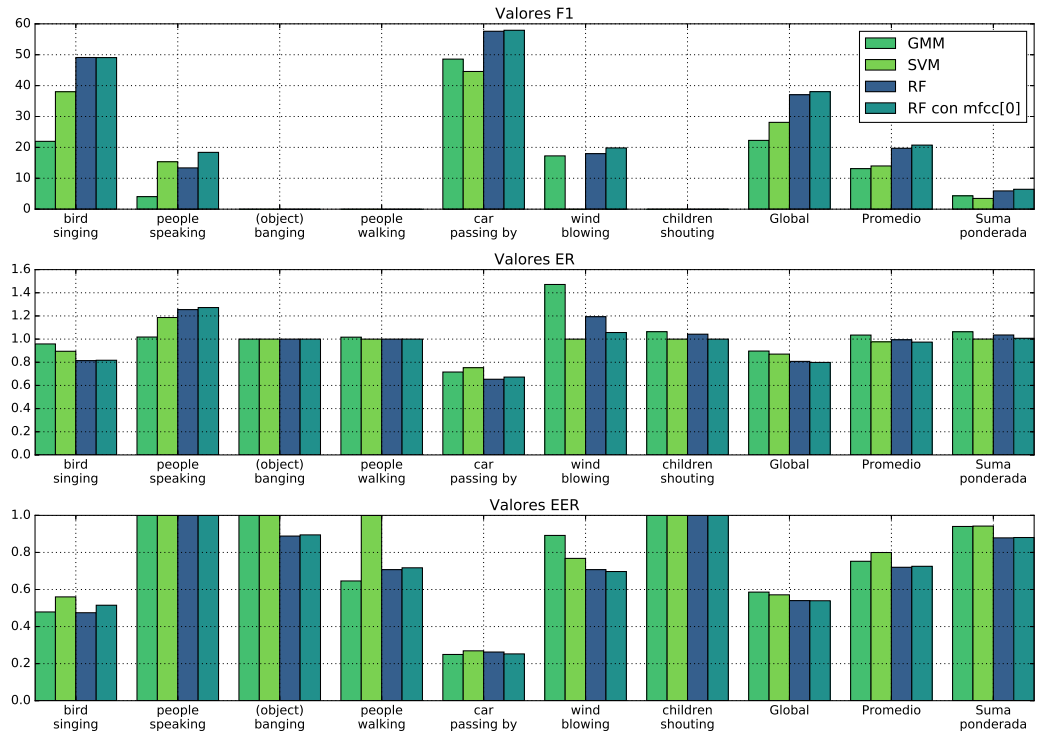


Figura 5.4: Resultados de $F1$, ER y EER para cada clase, globales, promedios y promedios ponderados para los clasificadores del *baseline*, SVM, Random Forest y Random Forest con MFCC[0].

a detectar mejor las clases mayoritarias. Esto se puede ver de forma concisa con los valores de la suma ponderada $\hat{F}1$ que son mucho más bajos que sus pares globales $F1$. En este sentido, se puede apreciar que aunque SVM obtiene un mejor desempeño global que el *baseline*, el resultado de la suma ponderada es peor, por lo que el *baseline* tiene más capacidad para detectar las clases minoritarias (PEOPLE SPEAKING en particular).

También se puede apreciar que los resultados de Random Forest son mejores para todas las medidas y agregar el coeficiente MFCC[0] ayuda a la detección. Esto se puede deber al hecho de que al agregar la información del coeficiente MFCC[0], que describe la energía de la señal, los árboles logren discriminar los casos donde hay silencio (sólo ruido ambiente), para no etiquetarlos. De esta forma se reduce el número de falsos positivos.

5.4. Edición de datos

Los resultados de las secciones anteriores muestran que el problema que se quiere atacar es difícil, principalmente debido a la superposición de eventos en el tiempo. Por otro lado, se puede ver que los resultados de todos los modelos

Sistema	ER	$F1(\%)$
Baseline	0.85	39.0
SVM	0.78	44.1
RF sin MFCC[0]	0.83	42.3
RF con MFCC[0]	0.75	48.8

Tabla 5.2: Comparación de resultados globales de la validación cruzada en el conjunto de *train* para los clasificadores *baseline*, SVM y Random Forest.

presentados no difieren de forma contundente. Por lo tanto, se puede pensar que los errores cometidos son debido a las propiedades de la base de datos. Por ejemplo, el gran desequilibrio de las clases y algunos errores de etiquetado.

Luego de examinar los datos del conjunto de *train* manualmente, se encuentra que los errores de etiquetado en general son eventos sonoros que no están etiquetados. El problema surge cuando se compara la salida del sistema con el *ground-truth*, donde hay varios segmentos que el sistema detecta como falsos positivos y en verdad no lo son. Para confirmar estos errores, se corrigen los errores de etiquetado y se re–entrenan y evalúan los diferentes modelos con validación cruzada. En la Tabla 5.3 se presentan los resultados obtenidos. El único algoritmo que no mejora su resultado al editar los datos manualmente es GMM con inicialización supervisada. Esto muestra el efecto nocivo de los errores de etiquetado en el desempeño de los algoritmos de clasificación.

	Sin edición de datos		Edición manual	
Sistema	ER	$F1(\%)$	ER	$F1(\%)$
Random Forest	0.75	48.8	0.67	53.1
SVM	0.78	44.1	0.70	51.5
GMM sup.	0.82	40.7	0.84	37.9
GMM *baseline*	0.85	39.0	0.79	42.5

Tabla 5.3: Resumen de resultados de la validación cruzada y comparación con y sin edición de datos

5.5. Tiempos de ejecución

Para comparar los tiempos de ejecución de los algoritmos, se corren en la infraestructura de cómputo de alto desempeño perteneciente a la Facultad de Ingeniería [67]. Para realizar una comparación fidedigna, se utilizan siempre 8 núcleos con 8 GB de RAM. En el caso de Random Forest se especifica que se paralelice el trabajo, ya que la implementación de *sckit-learn* lo permite.

En la Tabla 5.4 se puede confirmar que Random Forest, además de arrojar buenos resultados de clasificación, es también muy rápido para entrenar y clasificar

Sistema	Tiempo de train (s)	Tiempo de test (s)	Paralelización
Random Forest	311	78.8	Si
GMM *baseline*	387	76.9	No
GMM sup.	3390	80.5	No
SVM	15500	3000	No

Tabla 5.4: Comparación de tiempos de ejecución utilizando la infraestructura de cómputo de alto desempeño perteneciente a la Facultad de Ingeniería [67].

las muestras.

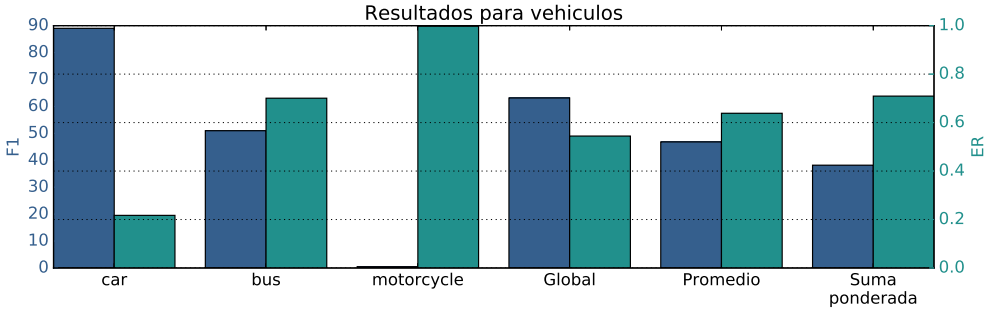
5.6. Resultados en SonidosMVD

En esta sección se prueba el algoritmo Random Forest en la base de datos SonidosMVD presentada en el Capítulo 2. Como se ve en la Figura 2.7, muchas de las clases nivel subordinado de etiquetas tienen muy pocas muestras, lo que puede dificultar el entrenamiento de los modelos. Por lo tanto, en primer lugar se entrenan modelos para las etiquetas de los niveles básicos (vehículos y componentes). En las Figuras 5.5a y 5.5b se muestran los resultados para vehículos y componentes respectivamente. Se puede ver que las clases MOTORCYCLE, BREAKS y COMPRESSOR casi nunca son detectadas. Además, aunque los resultados globales para ambos niveles son similares, las sumas ponderadas dan mejor para el nivel de vehículos. Esto se debe a que el sistema detecta mejor las clases mayoritarias en la categoría vehículos (CAR y BUS) que en la categoría de componentes (WHEEL y ENGINE).

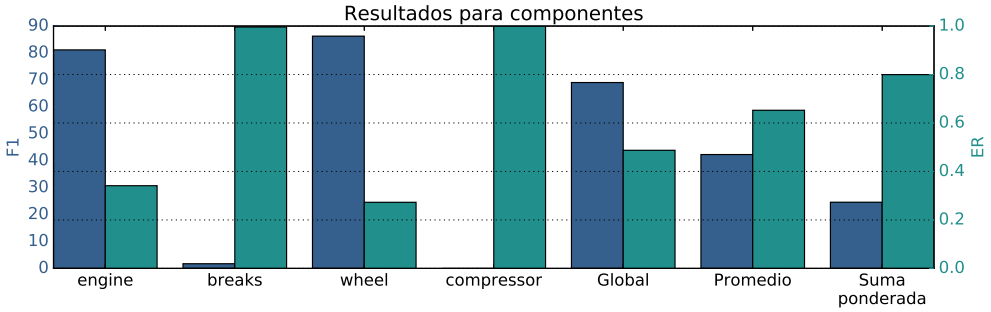
Los resultados en el nivel subordinado de etiquetas se muestran en la Figura 5.5c. Se puede ver que al aumentar el nivel de detalle de las etiquetas, el resultado de clasificación es peor, tanto para los valores globales como para las sumas ponderadas. En particular hay varias clases que nunca son detectadas. A continuación se intenta mejorar este resultado, variando el punto de trabajo.

5.6.1. Elección de un punto de trabajo

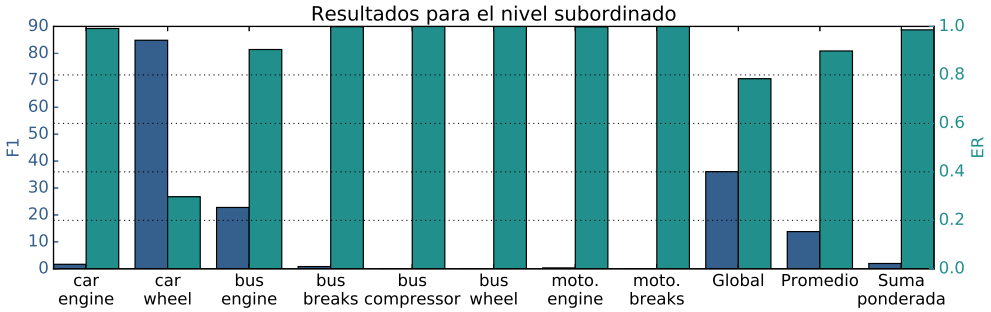
Como se ve en el Capítulo 3, la elección de las medidas de desempeño tiene gran importancia para la selección de un punto de trabajo en los sistemas diseñados. Elegir un punto de trabajo donde se maximice el $F1$ global puede llevar a detectar únicamente las clases mayoritarias. Por otro lado, maximizar $\hat{F}1$ puede llevar a mejorar la detección de las clases minoritarias, pero a costa de cometer más errores. Para confirmar esto, se plantean tres puntos de trabajo: (1) donde se maximiza la medida global $F1$; (2) donde se maximiza la suma ponderada $\hat{F}1$; y (3) donde se maximiza la relación $\hat{F}1/\hat{ER}$. La búsqueda de estos puntos, se realiza variando el umbral de detección γ y se evalúan los resultados en el conjunto de *validate*. En la Figura 5.6 se encuentran los resultados comparativos de los puntos de trabajo



(a) Categorías de vehículos.



(b) Categorías de componentes.

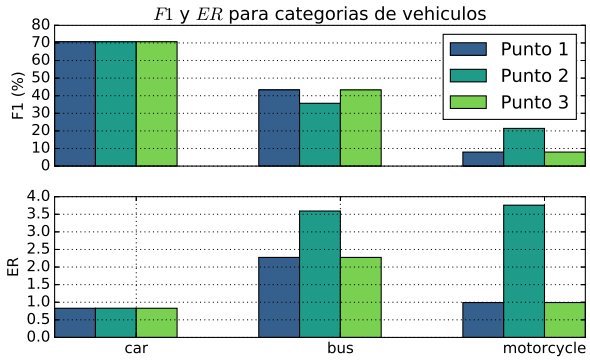


(c) Categorías del nivel subordinado.

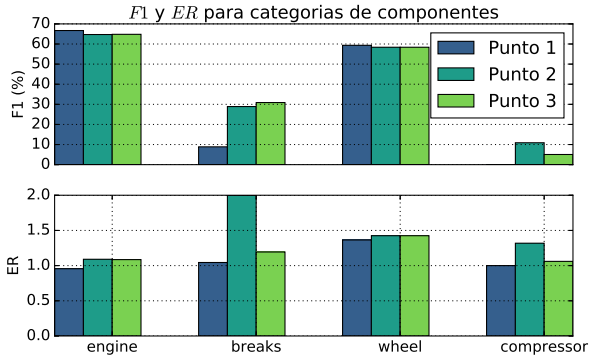
Figura 5.5: Resultados de $F1$ y ER por clase, globales, promedios y suma ponderadas en el conjunto de *test* de la base SonidosMVD para las categorías de (a) vehículos, (b) componentes y (c) del nivel subordinado.

para las tres categorías de clases en el conjunto de *test*.

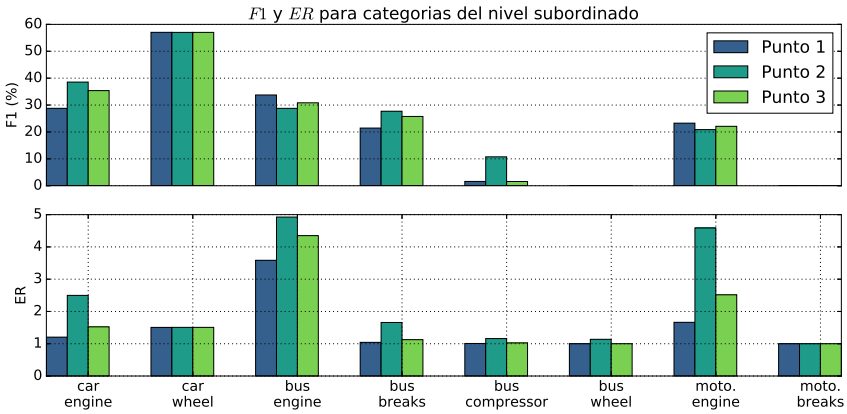
Notar que para los tres niveles de etiquetas se cumple que al maximizar $F1$ se tiende a detectar sólo las clases mayoritarias. Por otro lado, al maximizar la suma ponderada, para algunas clases mayoritarias se obtienen peores resultados, pero para varias de las minoritarias se mejora. Incluso se tienen clases que no se detectan maximizando la medida global y sí se detectan en algunos casos maximizando la suma ponderada. También notar que el error cometido es mayor. Un punto intermedio es el tercero, que busca maximizar $\hat{F1}$, pero sin aumentar mucho el



(a) Categorías de vehículos.



(b) Categorías de componentes.



(c) Categorías combinadas.

Figura 5.6: Resultados de $F1$ y ER por clase en el conjunto de *test* de la base SonidosMVD para las categorías de (a) vehículos, (b) componentes y (c) algunas del nivel subordinado. Se comparan tres puntos de trabajo: (1) donde se maximiza la medida global $F1$; (2) donde se maximiza la suma ponderada $\hat{F}1$; y (3) donde se maximiza $\hat{F}1/\hat{E}R$.

número de errores.

5.7. Discusión

En este capítulo se muestran los resultados de aplicar diferentes técnicas de reconocimiento de patrones en la base de datos TUT de eventos sonoros en áreas residenciales. Luego de ajustar los parámetros necesarios, se encuentra que GMM con inicialización supervisada, Random Forest y SVM, dan mejores resultados que en el *baseline* en el conjunto *train*. Por otro lado, los resultados en el conjunto de *test* muestran que GMM con inicialización supervisada tiende a sobre–entrenarse.

Se discute la utilidad de MFCC[0] como característica, ya que en el *baseline* no se utiliza con el objetivo de independizar la clasificación de la potencia de la señal y porque el resultado al incluirlo es peor (ER de 0.85 a 0.88). Esta característica resulta ser de utilidad cuando se utilizan otras técnicas como Random Forest o SVM. Esto puede deberse a que los clasificadores confunden los instantes de baja potencia con cualquier otra clase, generando falsos positivos.

Se comparan los tiempos de ejecución de los algoritmos, encontrando que Random Forest es el más rápido. Esto se debe, entre otras cosas, a que la implementación de *sklearn* permite paralelizar el entrenamiento de los árboles, aprovechando mejor los recursos.

Es interesante comparar los resultados obtenidos con Random Forest, con los de aquellos trabajos que obtuvieron los mejores desempeños en DCASE 2016. La comparación se hace sobre los resultados de la validación cruzada en el conjunto de *train* presentados en los artículos (Tabla 5.5). Se puede ver que los resultados obtenidos para el algoritmo de Random Forest son comparables a los reportados en DCASE 2016.

Sistema	ER	$F1(\%)$
Random Forest	0.75	48.8
Gorin [46]	0.75	51.5
Adavanne [44]	0.79	42.1
Baseline [22]	0.85	39.0

Tabla 5.5: Comparación de resultados con los presentados en DCASE 2016

A lo largo del capítulo, se confirma que la detección de eventos sonoros es un problema difícil de resolver. En particular, se tiene una base de datos relativamente chica y muy desbalanceada, lo que hace que algunas clases nunca sean reconocidas.

Por último, se prueba entrenar Random Forest con la base de datos SonidosMVD. Se realizan diferentes experimentos para los tres niveles de clases (vehículos, componentes y subordinado) que indican que, debido el desequilibrio de los datos, el sistema tiende a detectar solamente las clases mayoritarias. Este inconveniente se puede atenuar, maximizando una medida de desempeño que pondera con más peso a las clases minoritarias. Un resultado más equilibrado se obtiene

al maximizar el valor de la suma ponderada $\hat{F}1$ presentada en el Capítulo 3. Evidentemente la elección de una medida de desempeño y la selección de un punto de trabajo están cargadas de subjetividad y deben estar relacionadas con el objetivo del sistema. Pero en el problema de detección de eventos sonoros, el objetivo difícilmente sea detectar sólo una clase. Por lo tanto, una medida desempeño ponderada por la cantidad de muestras de cada clase, puede ayudar a encontrar un punto de trabajo más adecuado.

Capítulo 6

Detección basada en aprendizaje profundo (deep learning)

> *"Los paradigmas obtienen su status como tales, debido a que tienen más éxito que sus competidores para resolver unos cuantos problemas que el grupo de profesionales ha llegado a reconocer como agudos. Sin embargo, el tener más éxito no quiere decir que tenga un éxito completo. [El] éxito de un paradigma es [una] promesa de éxito discernible en ejemplos seleccionados y todavía incompletos."*
>
> – Tomas Kuhn, Estructura de las revoluciones científicas

En este capítulo se estudian las técnicas de *deep learning* aplicadas al problema de detección de eventos sonoros. Aunque el concepto de *deep learning* refiere a un conjunto muy variado de técnicas, en este trabajo el foco se pone en modelos discriminativos que incluyen transformaciones no lineales de los datos de entrada y que se entrenan junto con el clasificador [68]. Gran parte del éxito de estas técnicas se debe a la disponibilidad de grandes bases de datos, como ImageNet [69] o AudioSet [29]. Por más información sobre aprendizaje puede ser útil el libro de Goodfellow et al. [70].

Con la utilización de las técnicas de aprendizaje profundo se ha logrado mejorar los resultados en diferentes áreas, como por ejemplo, visión por computadora, reconocimiento de hablante, procesamiento de lenguaje natural, entre muchos otros [70]. Por lo tanto, dichas técnicas parecen ser reconocidas por gran parte de la comunidad de investigadores, como el nuevo estado del arte. Este fenómeno no es ajeno al área de interés de este trabajo, ya que la gran mayoría de los artículos publicados recientemente sobre detección de eventos sonoros incluyen técnicas de *deep learning*.

En este capítulo, se hace una introducción de diferentes arquitecturas, se comentan brevemente trabajos que utilizan dichos modelos y se demuestran los resultados de experimentos con las bases de datos URBAN-SED y SonidosMVD.

6.1. Perceptron multicapa

La red neuronal más simple es el perceptrón multicapa (*Multi Layer Perceptron*, MLP), basado en el modelo de neurona de Rossenblatt [71]. Consiste en la composición de funciones afines y activaciones no lineales intercaladas. Se denomina capa a cada par de función afín y activación no lineal. Entonces para cada capa i, se puede definir:

$$f_i\left(\mathbf{x}; \mathbf{W}_i, \mathbf{b}_i\right) = \rho_i\left(\mathbf{W}_i^T \mathbf{x} + \mathbf{b}_i\right) \tag{6.1}$$

donde $\mathbf{x}$ es la entrada, $\mathbf{W}_i \in \mathbb{R}^{d_{i-1} \times d_i}$ se denominan los pesos de cada capa, $\mathbf{b}_i \in \mathbb{R}^{d_i}$ son los sesgos (*bias*) y ρ_i es la función de activación [68, 70]. Se dice que cada capa i tiene dimensión d_i y que cada función f_i mapea el espacio $\mathbb{R}^{d_{i-1}}$ en $\mathbb{R}^{d_i}$.

Para obtener un modelo MLP de m capas, se deben concatenar las funciones f_i, por lo que para una entrada $\mathbf{x}$, la salida del modelo es:

$$f\left(\mathbf{x}; \theta\right) = \left(f_1 \circ f_2 \circ \cdots f_m\right)\left(\mathbf{x}\right) \tag{6.2}$$

donde $\theta = \left(\mathbf{W}_i, \mathbf{b}_i, \rho_i\right)_{i=1}^m$ es el conjunto de parámetros del modelo.

La función de activación de la última capa se elige en función de la estructura de la salida del modelo. En los problemas multi–etiqueta, como el de detección de eventos sonoros, suele utilizarse la función sigmoide, ya que puede interpretarse como la verosimilitud para cada etiqueta dado el dato de entrada. En este caso, las etiquetas se definen como un vector binario $y \in \{0, 1\}^C$, donde C es el número de clases [68].

Luego, se debe definir una función de costo, o de error, entre la salida del modelo y la etiqueta y para cada entrada $\mathbf{x}$, $f_{loss}(\mathbf{x}, y; \theta)$. La función de costo que se suele elegir para este problema es la *binary cross entropy* definida por

$$f_{loss}(\hat{y}, y) = \sum_{c=1}^{C} -y_c \log \hat{y_c} - (1 - y_c) \log(1 - \hat{y_c}), \tag{6.3}$$

donde $\hat{y} = f\left(\mathbf{x}; \theta\right)$ es la salida del modelo MLP. Esta función de pérdida es simplemente la suma de las log-verosimilitudes para cada etiqueta [70].

6.1.1. Entrenamiento

En general los parámetros del modelo se estiman por una variación del método del descenso por el gradiente. El algoritmo de descenso por el gradiente iterativo se puede describir de la siguiente forma:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta f_{loss}(\mathbf{x}, y; \theta^{(t)}) \tag{6.4}$$

donde $\eta > 0$ es conocido como el *learning rate* y ∇_θ denota el gradiente con respecto a los parámetros θ [68, 72]. Como la función f_{loss} es la composición de una serie de funciones simples, se puede calcular su gradiente utilizando la regla

de la cadena. Este método se denomina *error backpropagation*, ya que el cálculo del gradiente se realiza enviando información del error en la última capa de la red hacia las primeras capas [72].

La función de costo se define para todo el conjunto de entrenamiento, por lo que para cada paso del proceso del algoritmo se deben usar todos los datos para calcular $\nabla_\theta f_{loss}(\mathbf{x}, y; \theta^{(t)})$. Para evitar este problema, se utilizan métodos estocásticos que calculan el gradiente sobre una muestra aleatoria (*batch*) del conjunto de entrenamiento, lo que se denomina descenso por gradiente estocástico [72].

Otras variantes muy utilizadas como métodos de entrenamiento son Adagrad [73], Adadelta [74] y Adam [75], que se adaptan a los cambios relativos de los parámetros, disminuyendo la dependencia del valor de η.

6.1.2. Regularización

Como forma de regularizar los parámetros de los modelos y evitar sobre–entrenar, se utilizan principalmente dos métodos: *early stopping* y *dropout*. *Dropout* es una forma muy eficiente de regularizar, ya que simplemente se *elimina* un porcentaje definido de unidades en las capas intermedias de la red durante el entrenamiento. *Early stopping* se refiere a evitar que el modelo se sobre–entrene cortando el proceso de entrenamiento prematuramente en algún momento en que el resultado en el conjunto de validación deje de mejorar.

6.1.3. Ejemplo de modelo MLP para SED

Un ejemplo de MLP aplicado al problema de detección de eventos sonoros es el publicado por Mesaros et al. [35]. En la Figura 6.1 se puede ver un diagrama de dicho modelo, denominado M-MLP, cuya entrada es un vector de dimensión (d_0) 200. Este vector está formado por el logaritmo de la energía en 40 bandas mel de 0 a 22050 Hz de 5 *frames* de audio (el *frame* central, dos hacia adelante y dos hacia atrás). El largo de la ventana de análisis es de 40 ms y los *frames* están solapados un 50 %.

La estructura del modelo M-MLP consiste en dos capas ocultas de 50 unidades cada una ($d_1 = d_2 = 50$) con 20 % de porcentaje de *dropout*. Como la base de datos que se utiliza es la URBAN-SED y tiene 10 clases, la última capa tiene dicha dimensión ($d_3 = 10$). La red es entrenada usando el optimizador Adam durante 200 épocas, con *learning rate* de 0.001 y usando el criterio de *early stopping* para evitar sobre–entrenamiento.

Notar que la red utiliza poca información de contexto (60 ms hacia atrás y 60 ms hacia adelante), y en caso de señales no estacionarias esto puede ser un problema, ya que para dos *frames* consecutivos la salida de la red puede ser muy diferente. Además, este tipo de arquitecturas no aprovecha la estructura implícita de las señales de audio en tiempo o frecuencia [68]. Para aprovechar esta estructura, se han utilizado redes convolucionales como las presentadas en las secciones siguientes.

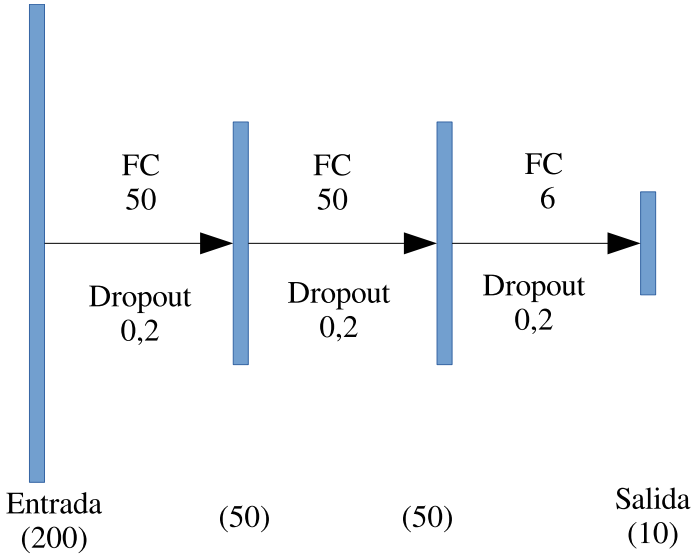


Figura 6.1: Arquitectura de modelo M-MLP de Mesaros et al. [35].

6.2. Redes convolucionales

En esta sección se estudian las redes convolucionales (CNN, Convolutional Neural Network) y se presenta un ejemplo de red que utiliza este tipo de arquitectura. Las redes convolucionales pueden ser unidimensionales (por ejemplo, para entradas de formas de onda de audio), bidimensionales (por ejemplo, para entradas de representaciones tiempo–frecuencia), o de mayor dimensión (por ejemplo, videos o múltiples representaciones tiempo–frecuencia). Esta sección se enfoca en redes de dos dimensiones ya que son muy utilizadas en el estado del arte del problema atacado en este trabajo. En la Sección 6.6 se estudian redes convolucionales de una dimensión.

La operación que define una capa de convolución es la siguiente:

$$f_i\left(\mathbf{X}; \mathbf{W}_i, \mathbf{b}_i\right) = \rho_i\left(\mathbf{W}_i * \mathbf{X} + \mathbf{b}_i\right), \tag{6.5}$$

donde $\mathbf{X} \in \mathbb{R}^{T_{i-1} \times F_{i-1} \times d_{i-1}}$ es la entrada de la capa, $\mathbf{W}_i \in \mathbb{R}^{n_i \times p_i \times d_{i-1} \times d_i}$ es el *kernel* de convolución y $\mathbf{b}_i \in \mathbb{R}^{d_i}$ es el *bias*. La salida de la función f_i pertenece al espacio $\mathbb{R}^{T_i \times F_i \times d_i}$. Notar que T_i y F_i son las dimensiones espaciales (tiempo y frecuencia respectivamente) de la entrada a la capa i; d_i es la cantidad de filtros de convolución en la capa i; n_i y p_i son las dimensiones de los núcleos de convolución en cada capa i [68, 70].

En caso de que la entrada del sistema sea un espectrograma, las dimensiones de entrada son: T_0 el número de *frames*; F_0 el número de bins de frecuencia; y $d_0 = 1$ en caso de señales monofónicas, $d_0 = 2$ para señales estereo, o $d_0 > 1$ si se utilizan entradas más complejas, como por ejemplo la combinación de diferentes representaciones tiempo–frecuencia.

En general, las operaciones de convolución son utilizadas en las primeras ca-

pas de la red y luego se concatenan capas denominadas totalmente conectadas[1], o densas, con la forma de la ecuación (6.1). Un ejemplo de red de este tipo es la presentada por Salamon et al. [19]. Esta red tiene tres capas de convolución seguidas de tres capas totalmente conectadas. La capa final es un sigmoide de 10 unidades que realiza la tarea de clasificación. Para lo que sigue se denomina S–CNN a esta arquitectura.

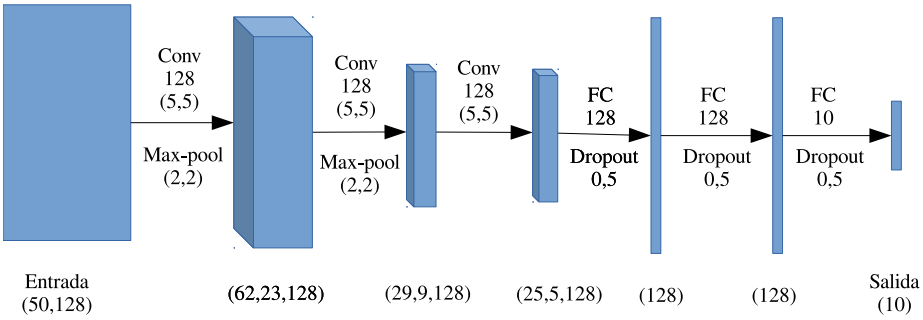


Figura 6.2: Arquitectura de la red S–CNN de Salamon et al. [19]

6.3. Redes recurrentes

Las redes recurrentes integran información temporal y se utilizan para modelar datos secuenciales. En su forma más simple, en una capa recurrente, se define el siguiente vector de estados $\mathbf{h}[t] \in \mathbb{R}^{d_{i-1}}$:

$$\mathbf{h}[t] = \rho\left(\mathbf{W}^T\mathbf{x}[t] + \mathbf{V}^T\mathbf{h}[t-1] + \mathbf{b}\right), \tag{6.6}$$

donde $\mathbf{x}[t] \in \mathbb{R}^{d_{i-1}}$ es la entrada de la capa, $\mathbf{W} \in \mathbb{R}^{d_{i-1}\times d_i}$ es la matriz de pesos de la entrada, $\mathbf{V} \in \mathbb{R}^{d_{i-1}\times d_i}$ se denomina matriz de pesos de la recurrencia y $\mathbf{b} \in \mathbb{R}^{d_i}$ es el vector de *bias* [68, 70].

La salida de la capa de recurrencia es la secuencia formada por los vectores de estado:

$$f_i\left(\mathbf{X}; \mathbf{W}_i, \mathbf{V}_i, \mathbf{b}_i\right) = \left(\mathbf{h}[t]\right)_{t=1}^{T}. \tag{6.7}$$

La función f_i mapea el espacio $\mathbb{R}^{d_{i-1}}$ en $\mathbb{R}^{d_i}$. En la práctica se utilizan unidades de recurrencia un poco más complejas como *gated recurrent unit* (GRU) [76] y *long short-term memory* (LSTM) [77].

Un ejemplo de red recurrente es la presentada por Adavanne et. al [44], que obtuvo el mejor resultado en el desafío de detección de eventos sonoros en grabaciones de zonas residenciales del DCASE 2016. La entrada de dicha red es el vector formado por la energía en 40 bandas mel. La red está formada por 2 capas ocultas con 32 unidades LSTM cada una, y una última capa con el número de unidades igual a la cantidad de clases. Se denomina esta red como A–RNN y en la Figura 6.3 se puede ver un diagrama de su estructura. Notar que a diferencia de la S–CNN

[1]Del inglés *fully connected*.

(ver Sección 6.2), la salida es una matriz de tamaño $(T, 10)$ donde para cada *frame* se tiene un vector de 10 activaciones. Por lo que, cada *frame* se clasifica según a qué clases pertenece, mientras que en la S–CNN dicha clasificación se hace para todos los *frames* conjuntamente. Básicamente la resolución de la clasificación de la A–RNN es a nivel de *frame* (23.2 ms), y en la S–CNN es de un segundo.

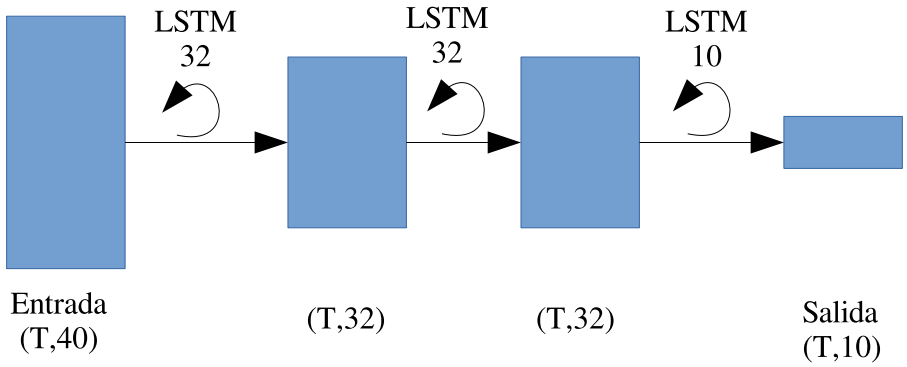


Figura 6.3: Arquitectura de la red A–RNN de Adavanne et. al [44].

6.4. Redes convolucionales - recurrentes

Una combinación interesante de las arquitecturas presentadas anteriormente son las *convolutional-recurrent neural networks* (CRNN) que combinan la extracción local de características (convolución) y su integración en el tiempo (recurrencia). Un ejemplo de este tipo de arquitectura es la presentada en [49], que incluye tres capas de convolución, tres capas de recurrencia y una capa totalmente conectada. Cada capa de convolución es seguida de *batch normalization*, *dropout* de 25 % y *max–pooling* en la dimensión de frecuencia. Este modelo se denomina C–CRNN y en la Figura 6.4 puede verse el diagrama de su arquitectura para un número T genérico de *frames*.

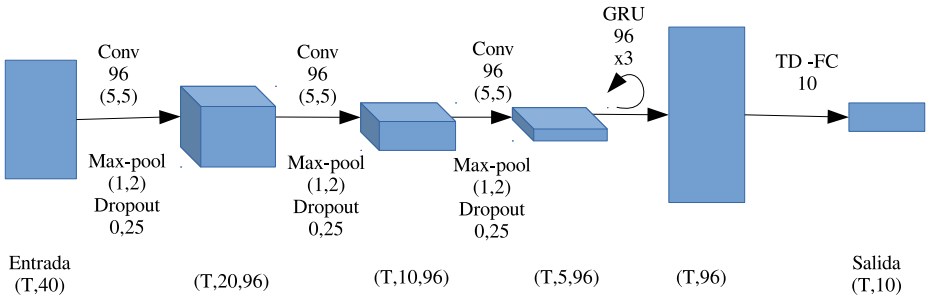


Figura 6.4: Arquitectura de la red C–CRNN de Cakir et al. [49]

6.5. Comparación de modelos

Se entrenan los modelos presentados en las secciones anteriores (M–MLP, A–RNN, S–CNN y C–CRNN) con la base de datos URBAN-SED de forma de comparar su desempeño. Para todos los modelos se utiliza la biblioteca *keras* [78]. En la Figura 6.5 se pueden ver las curvas ER vs. $F1$ en el conjunto de validación para los cuatro modelos. Un bueno modelo incluye valores altos de $F1$ y bajos de ER, zona asociada a la esquina superior derecha de la figura. Además, en la Tabla 6.1 se pueden ver los valores de ER y $F1$ en el punto de trabajo elegido (máximo $F1$) y para el conjunto de test. Se puede ver que con las redes que incluyen capas de convolución se obtienen mejores resultados, como era previsible. En la Figura 6.6 se puede ver que con S–CNN se obtienen mejores los resultados para casi todas las clases. Además entre las redes S–CNN y C–CRNN no hay mucha diferencia en el desempeño, y dado que la primera demora menos en entrenar, en lo que sigue se trabaja con esta.

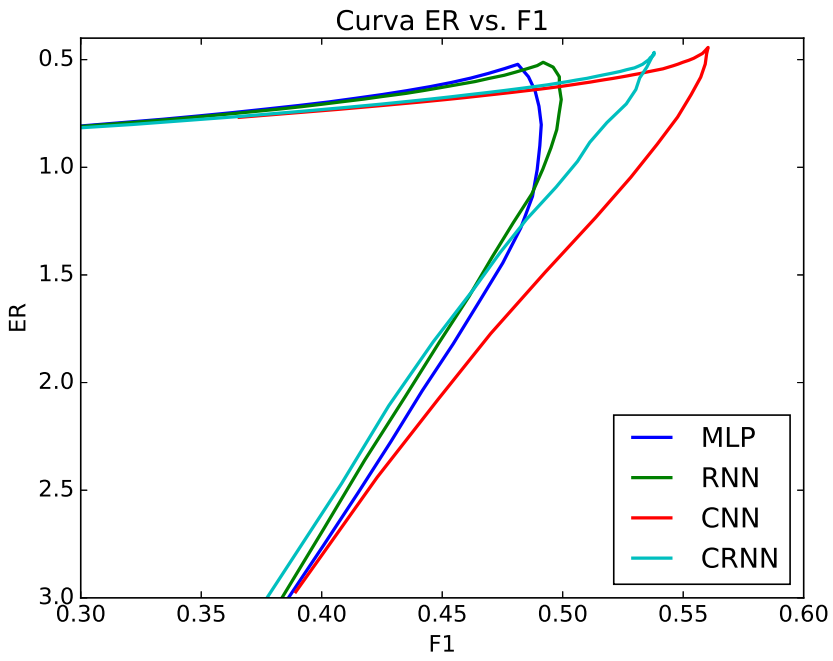


Figura 6.5: Curva ER vs. $F1$ para los modelos M–MLP, A–RNN, S–CNN y C–CRNN. Para generar esta curva se varía el umbral de decisión en la última capa de la red.

6.6. Redes punta–punta

Llamamos redes punta–punta (*end–to–end*) a aquellos modelos cuya entrada es la forma de onda de la señal de audio y la salida es el vector de clasificación. Este tipo de redes ha tenido excelentes resultados en clasificación de imágenes

Red	ER	$F1(\%)$
M–MLP	0.58	47
A–RNN	0.55	48
S–CNN	0.55	54
C–CRNN	0.53	53

Tabla 6.1: Resultados de $F1$ y ER en el punto de trabajo elegido dado por el máximo de F1, para los modelos M–MLP, A–RNN, S–CNN y C–CRNN en el conjunto de test.

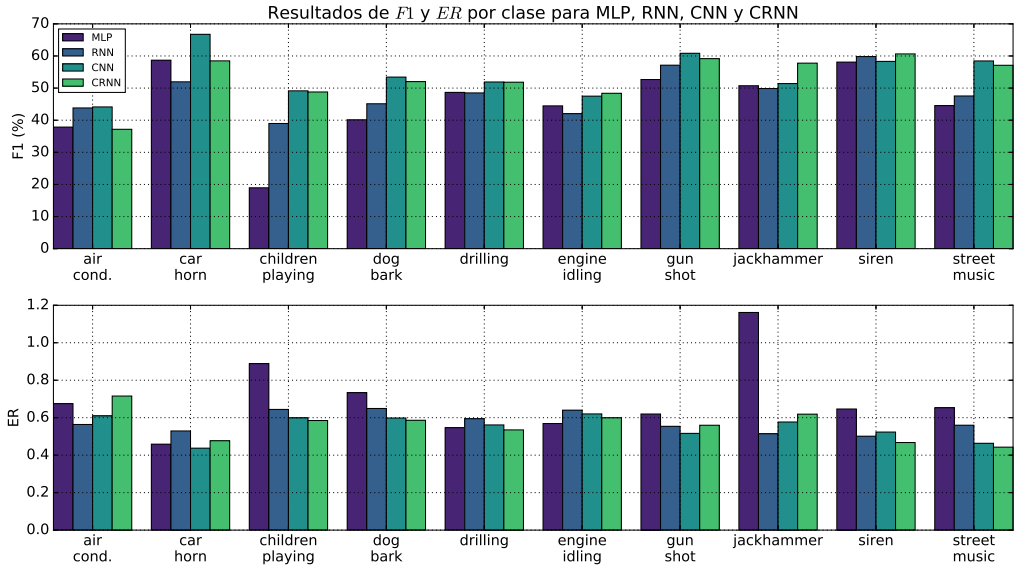


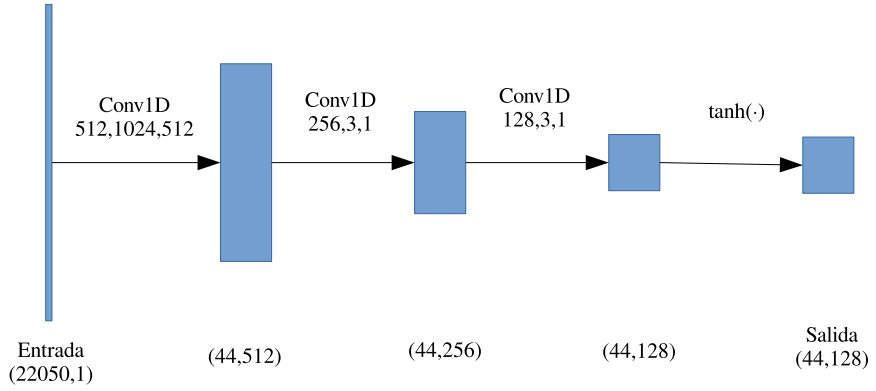
Figura 6.6: Curva de $F1$ y ER por clase para los modelos M–MLP, A–RNN, S–CNN y C–CRNN.

(la entrada es la imagen y la salida la clasificación), como por ejemplo AlexNet [62], VGG [79], GoogLeNet [80]; aunque en el dominio de las señales de audio todavía no se puede afirmar que obtengan mejores resultado que las redes cuya entrada es una representación de tiempo–frecuencia [65]. Redes de este tipo han sido utilizadas en problemas de detección del habla [81–83], reconocimiento de hablante [84], etiquetado automático de música [64,85] y transcripción automática de notas musicales [63].

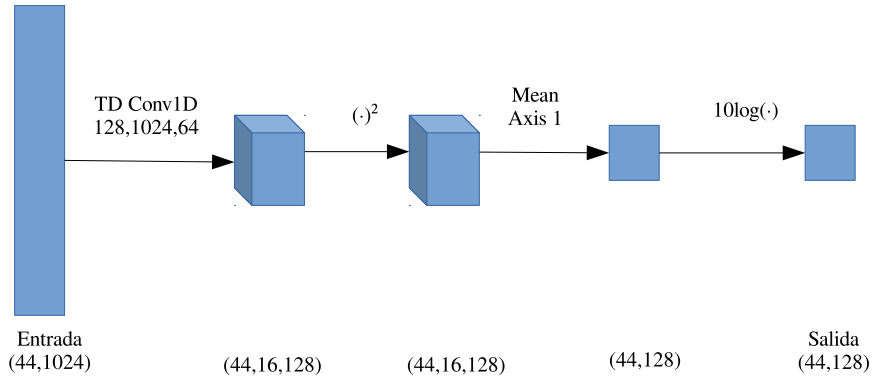
En las redes *end–to–end* la extracción de características usualmente se realiza en las primeras capas de convolución. Generalmente, estos modelos se utilizan como "cajas negras", esperando que la propia red "aprenda" las características acústicas que mejor discriminan las clases de interés a partir de los datos de entrenamiento. Sin embargo, es posible utilizar conocimiento del dominio para diseñar la etapa de extracción de características para un problema particular. El modelo MST (*mel–spectrogram transformation*) es un ejemplo de este enfoque [65], ya que su entrada es un segundo de la señal de audio y la salida es logaritmo del espectrograma en

escala mel. Si se concatena este modelo a una de las redes estudiadas anteriormente (por ejemplo S–CNN o C–CRNN) se obtiene una red *end–to–end*. Las primeras capas de la red pueden ser entrenadas para adaptarse a un problema específico, pero comenzando con una condición inicial que ha probado ser efectiva para este problema. Así, el proceso de entrenamiento puede requerir menor cantidad de datos o menos épocas.

En lo que sigue, se presenta el modelo MST, se propone un modelo alternativo (SMel) basándose en un banco de filtros simple, y se comparan ambos modelos concatenándolos con la red S–CNN presentada anteriormente (ver Sección 6.2).



(a) Modelo MST.



(b) Modelo SMel.

Figura 6.7: Diagrama de bloques de (a) modelo MST y (b) el modelo propuesto con $N_{mels} = 128$, $f_s = 22050$, ventana de 1024 puntos y salto de 512 puntos. Los parámetros mostrados en las capas de convolución son el número de filtros, el tamaño del *kernel* y valor de *stride* en ese orden. Encima de las flechas se muestran las dimensiones de las señales.

6.6.1. Modelo MST

El modelo MST, recientemente publicado, está basado en una red neuronal convolucional cuya entrada es un fragmento de señal de audio de un segundo de duración y la salida es el log–mel–espectrograma [65]. La arquitectura de la red incluye tres capas de convolución de 512, 256 y N_{mels} filtros respectivamente, donde N_{mels} es el número de bandas mel. En la Figura 6.7a se muestra el diagrama de la red para $N_{mels} = 128$, largo de ventana de 1024 puntos, *hop* de 512 puntos, y frecuencia de muestreo $f_s = 22050$ Hz.

6.6.2. Modelo propuesto (SMel)

En este trabajo se propone un enfoque más simple, un modelo denominado SMel, basado en que los pasos necesarios para calcular la energía en las bandas mel son filtros y otras funciones diferenciales que pueden ser implementadas como capas de una red neuronal. La entrada de SMel es una matriz cuyas columnas son los *frames* de la señal de audio. Cada *frame* (columna) es multiplicado por una ventana de Hann. La primera capa de la red es una convolución distribuida en el tiempo[2] de N_{mels} filtros y es inicializada con un banco de filtros mel. Entonces, la salida de la primera capa es el resultado de filtrar los *frames* de la señal con el banco de filtros. En las siguientes capas se calcula la energía en cada banda a través de una función para elevar al cuadrado; una función para promediar; y aplicar el logaritmo para convertir los valores de energía a decibeles (ver Figura 6.7b).

El banco de filtros mel está formado por N_{mels} filtros con respuesta en frecuencia triangular centrados en las frecuencias de la escala mel y solapados por la mitad del ancho de banda (ver Figura 4.1). Por lo que, la respuesta en frecuencia del filtro l es:

$$H_l(f) = \Lambda\left(\frac{f - f_l}{\Delta f_l}\right) \text{ para } f \geq 0, \tag{6.8}$$

donde f_l es la frecuencia central del filtro y $\Delta f_l = f_{l+1} - f_l$ es medio ancho de banda. Se diseñan respuestas al impulso para cada filtro como se muestra a continuación:

$$h_l(t) = 2\Delta f_l \text{sinc}^2(t\Delta f_l)\cos(2\pi f_l t)w(t), \tag{6.9}$$

donde $w(t)$ es una ventana de Hann.

6.6.3. Comparación de modelos

Para comparar ambos modelos, se los entrena con los mismos parámetros y con los mismos datos. También en este caso, la base de datos utilizada es la URBAN-SED.

[2]Capa que calcula el producto convolución para cada instante de tiempo de la señal de entrada. Estos instantes de tiempo son representados como una dimensión más de la señal [70].

El largo de la ventana se elige en $N = 1024$ muestras y el largo del salto se selecciona como la mitad. Se submuetrean los archivos de audio a una frecuencia de muestreo de 22050 Hz. La función objetivo (*ground truth* del mel–espectrograma) es calculada usando funciones de la biblioteca *librosa* con 128 bandas mel de 0 Hz a 11025 Hz.

Para entrenar el modelo MST, se usa la misma estrategia que en [65]. En cambio, para entrenar el modelo SMel es importante escoger con cuidado el *learning rate* porque la función logarítmica tiene un gradiente muy grande cerca del cero. Se usa la ecuación del descenso por el gradiente para estimar el *learning rate* en el peor caso.

Como la función de *librosa*, que es usada para convertir a decibeles, satura a -100 dB (potencia igual a 10^{-10}), consideramos este valor como el peor caso. Entonces, se estima el *learning rate* para tener pequeños cambios relativos en $x = 10^{-10}$.

Se entrenan ambos modelos durante 100 épocas utilizando el optimizador Adam y una función de *loss* cuadrática media. La variación del valor de la función de *loss* para cada modelo se muestra en la Figura 6.8. Es claro que la aproximación del modelo SMel es mejor que la del modelo MST en términos del valor de la función de loss alcanzado. Además, debido a que los filtros son inicializados antes del entrenamiento, la convergencia del modelo propuesto es más rápida. La Figura 6.9 muestra la salida para cada modelo para un archivo escogido al azar del conjunto de validación. Notar que la salida del modelo MST es más borrosa, principalmente en altas frecuencias.

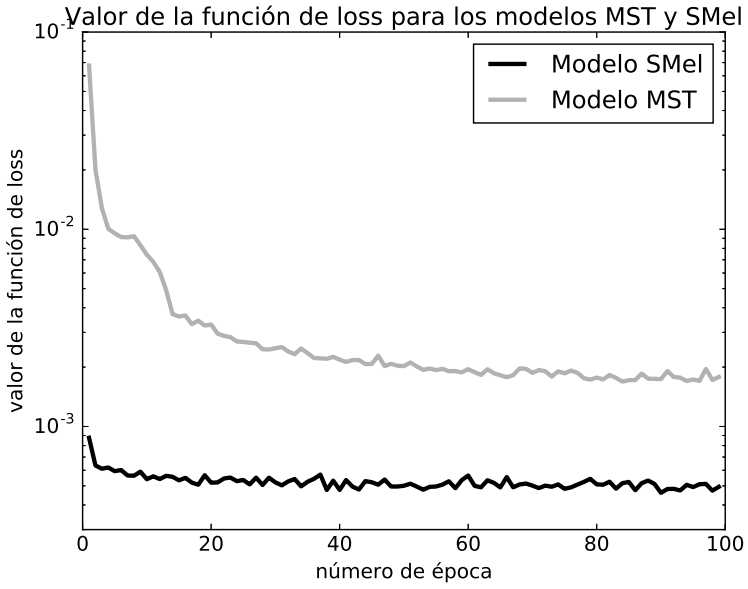


Figura 6.8: Variación del valor de la función de loss para cada modelo.

Los modelos de extracción de características (MST y SMel) se concatenan a la red convolucional S–CNN de la Sección 6.2 con el objetivo de generar redes

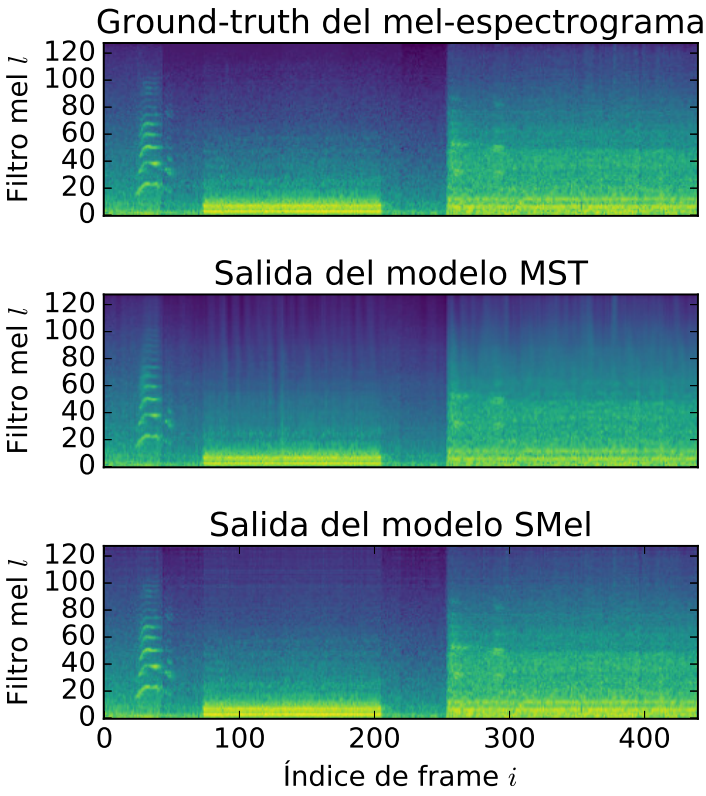


Figura 6.9: Salidas del ground-truth de librosa, del modelo MST y el modelo propuesto para un archivo del conjunto de validación elegido al azar.

end–to–end. Entonces se comparan tres redes: (S–CNN) el trabajo de base (con mel–espetrograma de *librosa* como entrada) [19]; (MST+S–CNN) el modelo MST concatenado con S–CNN; y (SMel+S–CNN) el modelo propuesto también concatenado con S–CNN.

6.6.4. Estrategia de entrenamiento

Para entrenar S–CNN, se usa la misma estrategia que en la Sección 6.2 y los parámetros definidos en la Sección anterior. Por otro lado, para entrenar MST+S–CNN y SMel+S–CNN, se usa una estrategia inspirada en la *Branch Training strategy* para clasificación jerárquica [86]. Las sub-redes S–CNN, MST y SMel se inicializan con los parámetros aprendidos en los experimentos anteriores. Se entrenan las redes concatenadas con dos funciones de *loss*: *mean squared* para la salida del log–mel–espectrograma (l_0) y *binary cross entropy* para la salida de clasificación (l_1). La función de *loss* total (l) es una suma ponderada de ambas funciones:

$$l = w_0 l_0 + w_1 l_1, \tag{6.10}$$

donde $[w_0, w_1]$ es el par de pesos. La función de loss l_0 funciona como una especie de regularización sobre la extracción del log–mel–espectrograma por lo que el peso w_0

define cuánto se regulariza. Los parámetros de las redes MST+S–CNN y SMel+S–CNN se inicializan con aquellos de las subredes que las componen (MST, SMel y S–CNN). Además, se cambia la implementación del optimizador Adam para permitir que el *learning rate* sea diferente en la subred de extracción del log–mel–espectrograma y en la subred de clasificación.

6.6.5. Resultados

Se entrenan las tres redes por 100 épocas usando las estrategias propuestas previamente y utilizando el optimizador Adam modificado. La Figura 6.10 muestra las variaciones del valor de $F1$ en el conjunto de validación para las tres redes. Se guardan los pesos de las redes en la época donde el valor de $F1$ es mayor. La Tabla 6.2 muestra los resultados de $F1$ para el mejor modelo de cada red. Se puede ver que SMel+S–CNN obtiene resultados levemente mejores que S–CNN y bastante mejores que MST+S–CNN.

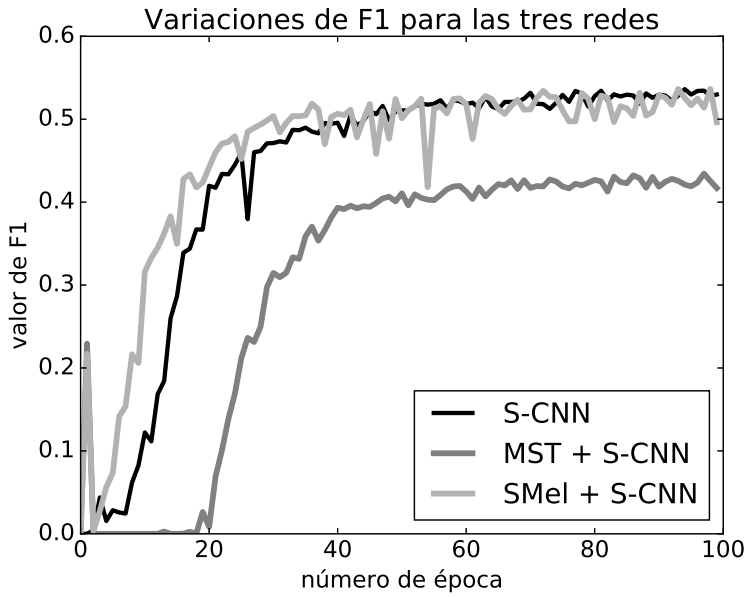


Figura 6.10: Variaciones del valor de F1 en el conjunto de validación para cada modelo.

Red	ER	$F1(\%)$
S–CNN	0.53	56
MST+S–CNN	0.61	43
SMel+S–CNN	0.50	57

Tabla 6.2: Resultados de ER y $F1$ en el conjunto de *test* para las redes S–CNN, MST+S–CNN y SMel+S–CNN.

6.6.6. Normalización con PCEN

Para estudiar el efecto de utilizar PCEN para normalizar el mel–espetrograma en el problema de detección de eventos sonoros, se plantea una simplificación de la red SMel+S–CNN de la parte anterior. Se define un modelo de extracción del mel–espetrograma, llamado SMel_P, que tiene como entrada el espectrograma de la señal. En este caso, para calcular la energía en las bandas mel sólo es necesario multiplicar la entrada del modelo por el banco de filtros mel en frecuencia ($H_l[k]$). Luego se implementan las operaciones necesarias para calcular PCEN (ver ecuación 4.7). Para calcular $M\,[i,l]$ se utiliza una celda recurrente diseñada específicamente para que implemente el filtro IIR propuesto [59]. Para el resto de la operación se diseña una capa específica que implementa PCEN que recibe dos entradas, $E\,[i,l]$ y $M\,[i,l]$. Los parámetros de la normalización se definen dependientes de la frecuencia, por lo que la capa diseñada que tiene el conjunto de parámetros $\alpha\,[l]$, $\delta\,[l]$, $r\,[l]$ y $b\,[l]$. En el diagrama de la Figura 6.11 se puede ver cómo está implementado el modelo con capas de una red neuronal.

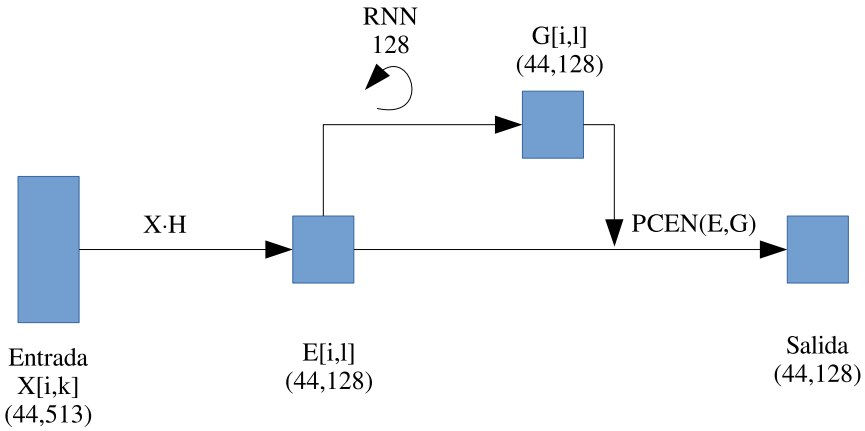


Figura 6.11: Arquitectura del modelo SMel_PCEN. Extracción de energía en las bandas mel normalizadas con PCEN.

De la misma forma que se hace anteriormente, se concatena el modelo SMel_P con la S–CNN para obtener un sistema de detección de eventos sonoros. En primer lugar se entrena la S–CNN con los datos de entrada normalizados con la función PCEN de *librosa* y los mismos parámetros de la Sección 4.3. Esta red se denomina S–CNN_P. Luego se entrena la red concatenada SMel_P+S–CNN_P con la misma función de loss de la ecuación (6.10). En este caso se quiere analizar cómo cambian los filtros $H_l[k]$ por lo que w_0 se fija en cero para que no haya regularización. Los parámetros de PCEN se inicializan con los mismos valores que en S–CNN_P. En la Figura 6.12 se puede ver cómo resultan luego de entrenar. El único parámetro que porcentualmente cambia considerablemente es r, que define cómo trabaja el compresor de rango dinámico (DRC). Para altas frecuencias, el valor de r disminuye, por lo que el compresor actúa con más fuerza [60]. Análogamente, la compresión es más suave para bajas frecuencias. Esto se puede deber a que la información más relevante para clasificar las fuentes sonoras se encuentra en las bajas frecuencias.

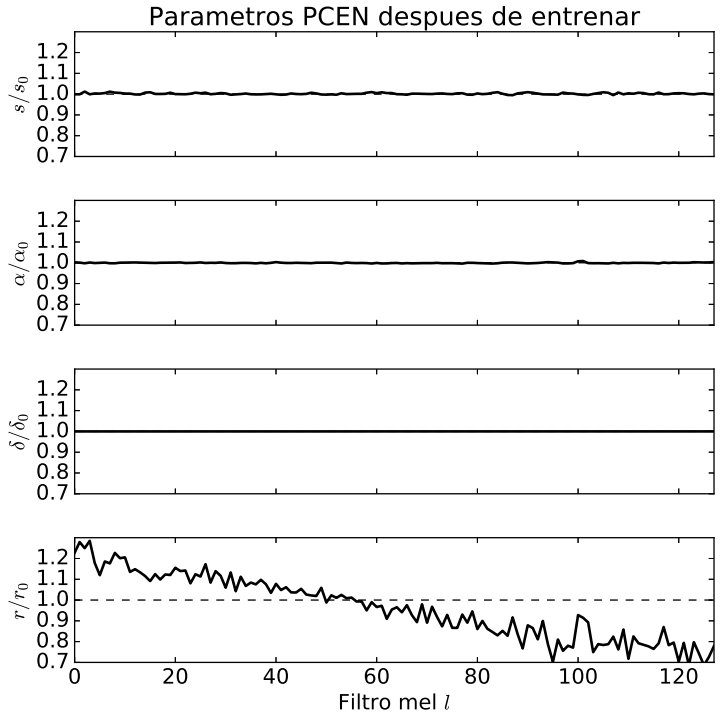


Figura 6.12: Parámetros de PCEN referenciados a los valores iniciales en función del canal (banda frecuencial l) entrenados con SMel_P+S–CNN_P. Con lineas punteadas se marcan los valores iniciales.

Red	ER	$F1(\%)$
S–CNN	0.53	56
S–CNN_P	0.56	54
SMel_P+S–CNN_P	0.60	51

Tabla 6.3: Resultados de $F1$ y ER en el conjunto de test para la redes S–CNN, S–CNN_P y SMel_P+S–CNN_P.

En la Tabla 6.3 se pueden ver los resultados del desempeño de las redes S–CNN, S–CNN_P y SMel_P+S–CNN_P. En primer lugar se puede ver que la normalización PCEN con los parámetros definidos no mejora el desempeño de S–CNN. Luego se ve que al entrenar la red SMel_P en conjunto con S–CNN_P tampoco se obtienen mejores resultados, pero es interesante ver en la Figura 6.13 cómo varían los filtros $H_l[k]$. Para las altas frecuencias el resultado es muy ruidoso, en particular a partir de 4000 Hz, y la información en esa banda de frecuencias puede no estar aportando información para la clasificación. Para confirmar esta idea, se prueba re-muestrear la base de datos a 8000 Hz y calcular la energía en 64 bandas mel en el rango de 0 a 4000 Hz. Se puede ver que efectivamente los resultados de ER y $F1$ no cambian significativamente trabajando a una frecuencia de muestreo de 8000 Hz, mientras que el número de parámetros se reduce notoriamente (ver Tabla 6.4).

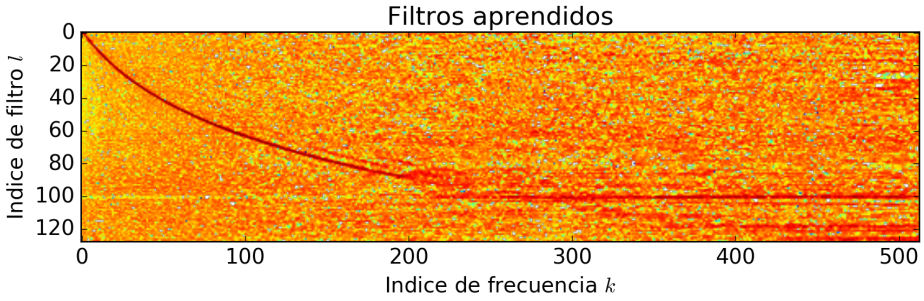


Figura 6.13: Filtros $H_l[k]$ aprendidos por la red SMel_P+S–CNN_P.

$f_s(Hz)$	Red	$F1(\%)$	ER	# params (M)
22050	S–CNN_P	54	0.56	~ 2.48
	SMel_P+S–CNN_P	51	0.60	~ 2.55
8000	S–CNN_P	52	0.55	~ 0.99
	SMel_P+S–CNN_P	49	0.56	~ 1.01

Tabla 6.4: Comparación de resultados de $F1$ y ER en el conjunto de test y número de parámetros de la redes S–CNN_P y SMel_P+S–CNN_P entrenadas con la base URBAN-SED remuestreada a 22050 Hz y 8000 Hz.

6.7. Experimentos con SonidosMVD

Para probar las técnicas de *deep learning* en las grabaciones de SonidosMVD, se aplica el enfoque de *fine tuning*. Se trata de aprovechar el entrenamiento de las redes en bases de datos de mayor tamaño. Se busca mantener la capacidad de las primeras capas para extraer características relevantes, mientras que la(s) última(s) capa(s) se re-entrena(n) para detectar los eventos de la nueva base de datos.

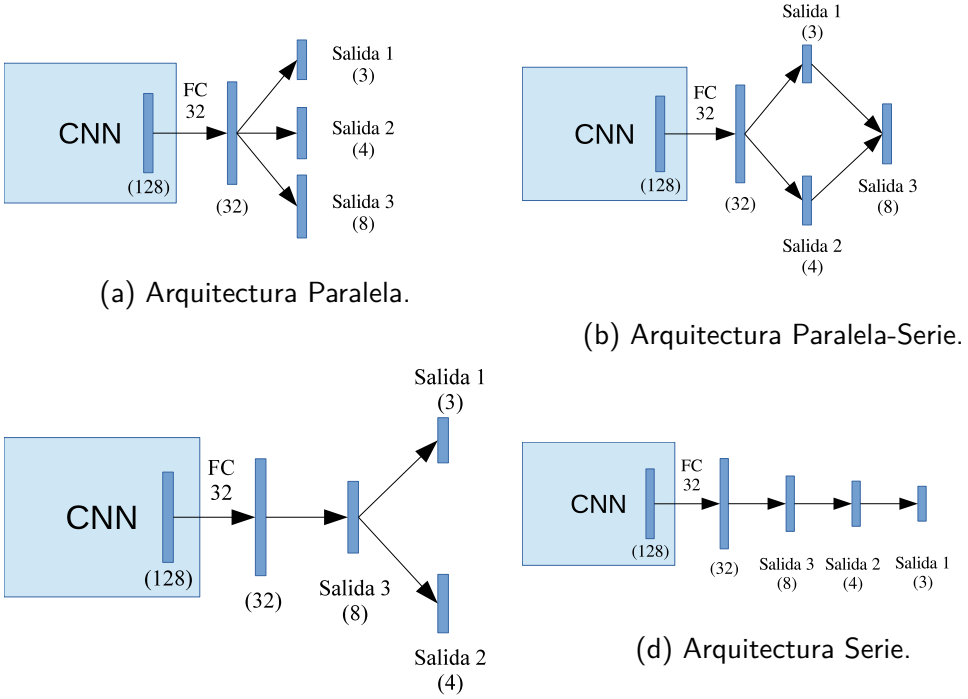
En este caso, se utiliza la red S–CNN entrenada como en la secciones anteriores, *quitando* la última capa de clasificación (FC 10, ver Figura 6.2). A esta red recortada, se le concatenan nuevas capas de clasificación para detectar eventos en los tres niveles de clases definidos en esta base; para esto se define una función de *loss* para cada nivel de clases. Se diseñan tres arquitecturas diferentes, que intentan aprovechar el diseño taxonómico de SonidosMVD (ver Figura 6.14). En todas ellas, se define una primera capa *fully-conected* que busca recibir información de todas las funciones de *loss*. Luego, las cuatro arquitecturas se definen de la siguiente manera.

(a) Paralela. Tiene tres salidas en paralelo para cada uno de los niveles de categorías. En este caso la relación entre las categorías sólo se puede modelar en la capa "FC 32".

(b) Paralela-Serie. Tiene dos salidas en paralelo para clasificar las categorías de vehículos y componentes. Esta salidas se concatenan para luego conectarse con la salida de clases del nivel subordinado.

(c) Serie-Paralela. Tiene una primera salida para las categorías combinadas que luego se conecta con dos salidas en paralelo para clasificar las clases de vehículos y componentes.

(d) Serie. Tiene las tres salidas concatenadas: nivel subordinado, categorías de componentes y categorías de vehículos en ese orden.



(a) Arquitectura Paralela.

(b) Arquitectura Paralela-Serie.

(c) Arquitectura Serie-Paralela.

(d) Arquitectura Serie.

Figura 6.14: Diagrama de bloques de las cuatro arquitecturas probadas para hacer *fine tuning*.

Los cuatro modelos se entrenan durante 200 épocas con optimizador Adam y un *learning rate* de 0.0002. Sólo se entrenan las capas agregadas en cada arquitectura. Se guardan los pesos de la red cuando se mejora el promedio de los valores globales de $F1$ de las tres salidas. Los resultados de $F1$ en el conjunto de validación para cada arquitectura y para cada categoría se presentan en la Figura 6.15. Notar que los resultados son mejores para la arquitectura serie, por lo que es utilizada en el resto del capítulo. En la Figura 6.16 se presentan los resultados por clase, los resultados globales y los promedios. Notar que la red tiende a reconocer solamente las clases mayoritarias, por lo que el resultado de la suma ponderada es bajo.

Hasta el momento se trabaja con redes S–CNN a las que se les ha cortado la última capa. Este enfoque puede ser sub–óptimo ya que la especialización de la red para los datos que fue entrenada, también puede darse en capas inferiores. Para estudiar esto, se comparan los resultados de (a) cortar en la última capa, (b) cortar en la penúltima y (c) cortar en la ante penúltima. En la Figura 6.17 se presentan

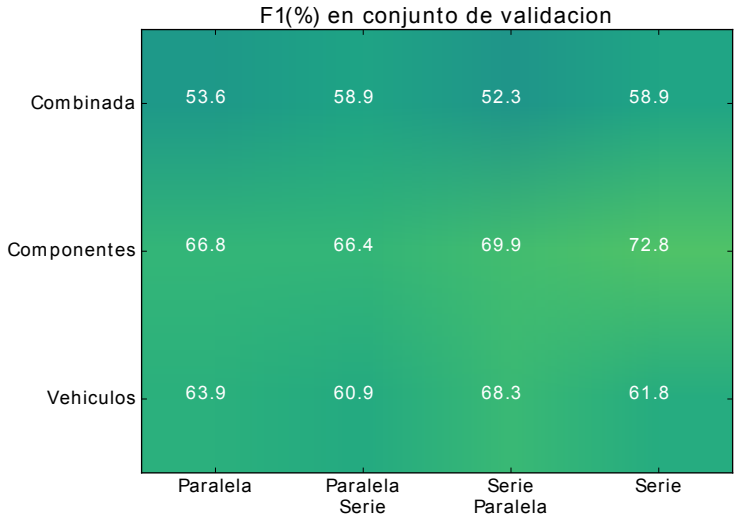
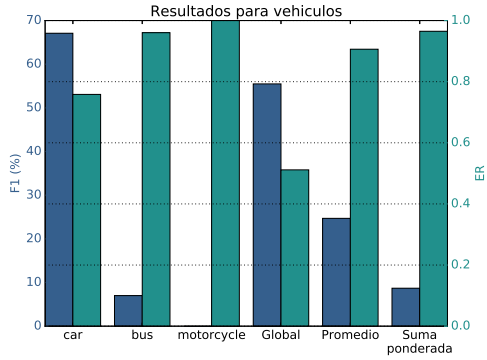
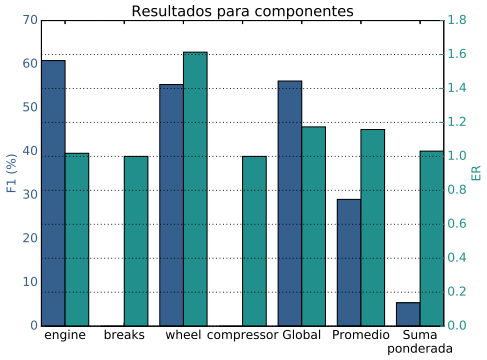


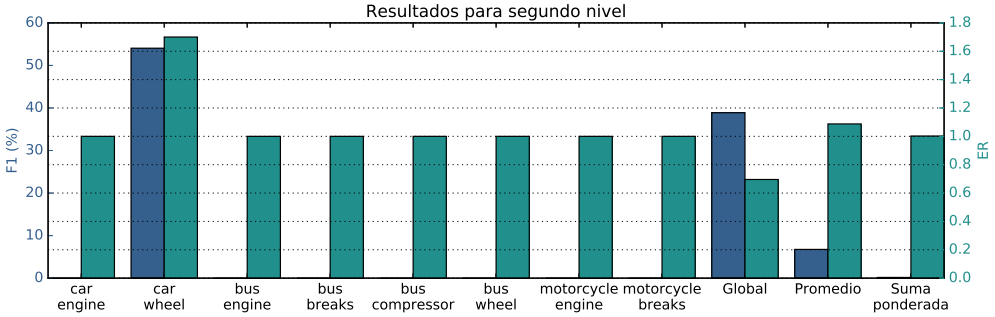
Figura 6.15: Resultados globales de $F1$ en el conjunto de validación para cada arquitectura y cada categoría de clases.



(a) Categorías de vehículos.



(b) Categorías de componentes.



(c) Categorías combinadas.

Figura 6.16: Resultados de $F1$ por clase en el conjunto de *test* de la base SonidosMVD para las categorías de (a) vehículos, (b) componentes y (c) algunas del segundo nivel en el conjuntos de *test* para la base SonidosMVD.

los resultados de considerar estas tres alternativas que indican que en este caso, lo mejor es cortar la red en la penúltima capa.

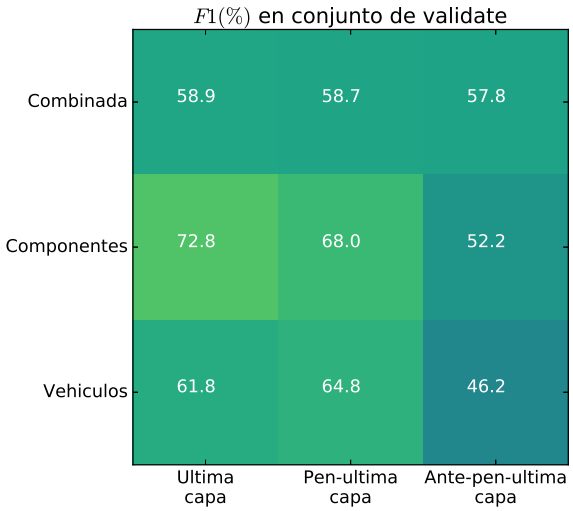


Figura 6.17: Comparación de resultados globales de $F1$ en los conjuntos de validate (izquierda) y test (derecha) para cada tipo de corte de la red S–CNN y para cada categoría de clases.

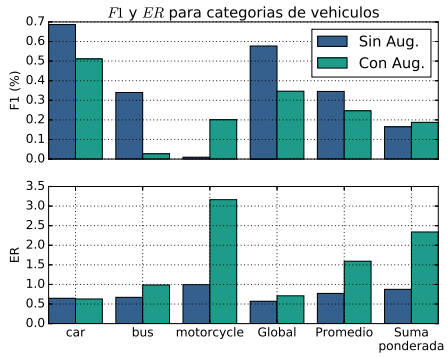
6.7.1. Data augmentation

Data augmentation refiere a un conjunto de técnicas para sobre–muestrear el conjunto de entrenamiento de forma de mejorar el desempeño para las clases minoritarias. Esta técnicas son muy utilizadas cuando se usa *deep learning* y han ayudado a mejorar el desempeño en diversos problemas. Se busca realizar transformaciones coherentes a los datos, de forma que mantengan la información relevante pero que no sean copias exactas. En el problema de detección de eventos sonoros, estas transformaciones se pueden realizar sobre varios eventos, y luego se pueden sumar para generar diferentes tipos de mezclas.

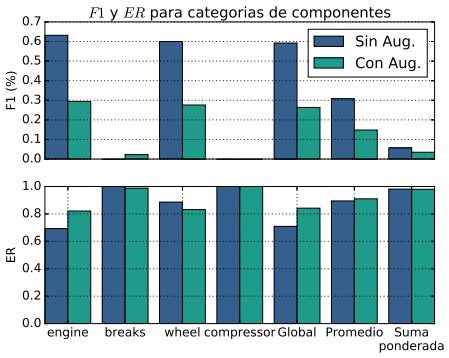
Para hacer este proceso, se utiliza la biblioteca *scaper* [19] para síntesis de entornos sonoros. Básicamente, la biblioteca permite agregar eventos a un fondo que puede ser silencio, ruido o cualquier otra señal de audio. En este caso, como señales de fondo se utilizan todos los archivos del conjunto de entrenamiento. Del mismo conjunto se extraen los eventos de las clases minoritarias. Luego, se utiliza la biblioteca para sintetizar tres entornos sonoros para cada uno de los archivos y se agregan eventos aleatorios de las clases minoritarias. Las transformaciones que se realizan a los eventos son: cambio de tono, cambio de velocidad (o duración) y cambio de intensidad.

En la Figura 6.18 se comparan los resultados de entrenar con los datos originales y con los generados con *scaper*. En ambos casos, en el entrenamiento, se busca maximizar el valor de $\hat{F}1/\hat{ER}$. Se puede ver que al agregar eventos se mejora el

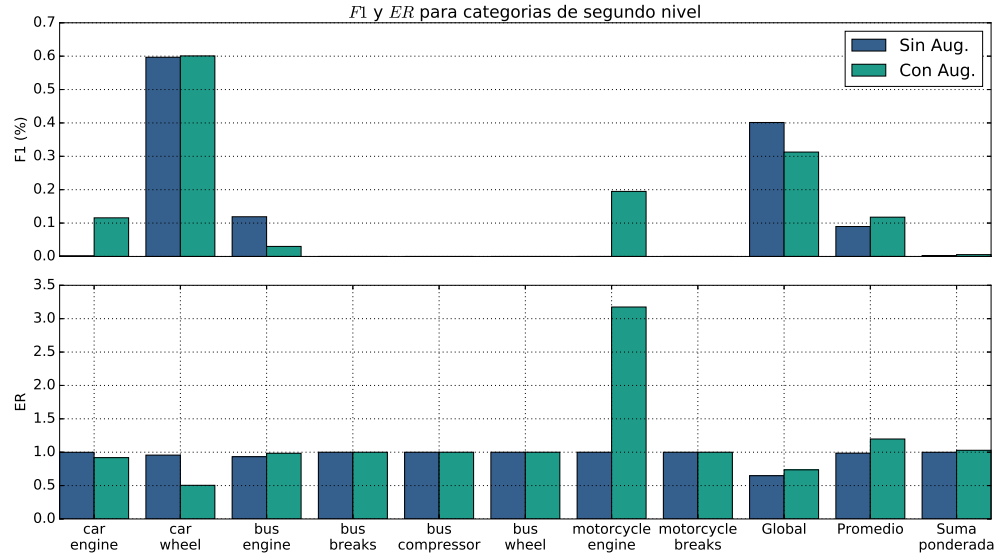
desempeño en las clases minoritarias, mientras que aumentan las tasas de error para la mayoría de las clases.



(a) Categorías de vehículos.



(b) Categorías de componentes.



(c) Categorías combinadas.

Figura 6.18: Resultados de $F1$ por clase en el conjunto de *test* de la base SonidosMVD con y sin *data augmentation* para las categorías de (a) vehículos, (b) componentes y (c) algunas del segundo nivel en el conjuntos de *test* para la base SonidosMVD.

6.8. Discusión

En este capítulo se presentan algunas arquitecturas para la detección de eventos sonoros basadas en *deep learning*. Con diferentes experimentos se muestra que las redes con capas convolucionales obtienen los mejores resultados.

Luego, se presenta un enfoque novedoso para la detección de eventos sonoros en entornos urbanos usando redes neuronales *end–to–end*, que se obtiene conca-

tenando dos redes: una para la extracción de características y otra para realizar la clasificación. Esta arquitectura de dos etapas facilita la introducción de conocimiento del dominio y mejora la interpretabilidad de lo que las redes aprenden. Para la primera red, que es diseñada para extraer el mel–espectrogram, se propone un modelo simple basado en un banco de filtros. Se muestra que el modelo propuesto obtiene mejor resultado (menor valor de función de *loss*) que el que se obtiene con el modelo MST. También se muestra que los resultados de clasificación de la red *end–to–end* son similares a los que se obtienen con las redes convolucionales del estado del arte. Sin embargo, con el modelo propuesto se facilita la interpretación de los parámetros aprendidos

Además, se implementa la normalización PCEN como una red neuronal y se entrenan sus parámetros en conjunto con el resto de la red. Se encuentra que el parámetro que cambia más significativamente en el entrenamiento es r, que determina cuánto afecta el compresor a la señal. Por otro lado, se estudian los filtros de la primera capa después de entrenados. Los resultados muestran que para la base de datos URBAN-SED, la información más relevante se encuentra por debajo de 4000 Hz y esto es confirmado remuestreando la base de datos a 8000 Hz. Se concluye que se pueden diseñar modelos con menor cantidad de parámetros.

Por último, se presentan resultados de aplicar *fine tuning* a la red S–CNN con los datos de SonidosMVD. Se observa que el óptimo se encuentra entrenando una arquitectura serie y cortando la red en la penúltima capa. Se encuentra que con estas técnicas de clasificación se detectan bien las clases mayoritarias, pero no se detectan las minoritarias. Para mejorar este resultado, se sintetizan entornos sonoros con la biblioteca *scaper* con el objetivo de sobre–muestrear las clases minoritarias. Se muestra que entrenando con los datos sintetizados, se detecta mejor las clases minoritarias, pero se cometen más errores. Para mejorar este resultado, seguramente es necesario generar una mayor cantidad de datos para tener mejor representadas las clases minoritarias.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

En esta tesis se estudian herramientas computacionales para la detección de eventos sonoros en entornos urbanos. Durante todo el trabajo se muestra que este es un problema complejo y desafiante. Las características acústicas de cada clase pueden tener una gran diversidad debido a la variabilidad entre fuentes sonoras de un mismo tipo (p. ej. autos) y la influencia del ambiente (p. ej. reverberación). Además, la superposición temporal de eventos sonoros también dificulta la clasificación.

Se necesita definir una metodología de trabajo que permita realizar una estimación correcta del desempeño de los algoritmos. En este sentido, se estudiaron las medidas de desempeño que han sido propuestas para este problema ($F1$ y ER) [23, 26, 35, 36] y se muestra que estas presentan algunos inconvenientes cuando los datos tienen mucho desequilibrio de clases. El problema abordado en condiciones reales implica necesariamente conjuntos de datos muy desbalanceados, por lo que es necesario definir medidas de desempeño más adecuadas. En varios experimentos se muestra que, entrenando los algoritmos con bases de datos muy desbalanceadas, se tiende a detectar solo las clases mayoritarias, lo que reafirma la necesidad de utilizar una medida de desempeño que penalice esta situación. Se propone un método para ponderar los valores de $F1$ y ER por clase. El peso de cada clase es inversamente proporcional a la cantidad de muestras de dicha clase, de forma tal que las categorías con menos datos tienen más peso en las medidas.

En esta tesis también se presentan las bases de datos disponibles para este problema: UrbanSound8k [17], URBAN-SED [19] y TUT [22]. Se muestran las ventajas y limitantes de estas bases de datos, y la necesidad de generar registros sonoros etiquetados de la ciudad de Montevideo. En este sentido, se exponen los procesos de diseño, grabación y etiquetado de la base de datos SonidosMVD. El diseño de SonidosMVD incluye la definición de las clases de interés y una taxonomía que describe cómo estas relacionan. Este trabajo se centra en ruidos molestos, en particular los del tránsito, y se opta por una taxonomía doble que describe tanto a los vehículos como a los componentes que generan el ruido.

Para la clasificación de los eventos sonoros se presentan dos enfoques: aprendizaje poco profundo y aprendizaje profundo. El primero refiere a aplicar técnicas de procesamiento de señales de audio para la extracción de características y la utilización de técnicas de reconocimiento de patrones para la clasificación. Para este enfoque se realizan distintos experimentos para comparar el desempeño de modelos GMM, SVM y Random Forest, utilizando MFCC como características. De los experimentos surge que con Random Forest se obtiene el mejor desempeño, incluso comparable a los resultados de los algoritmos ganadores del desafío DCASE 2016. En un punto de trabajo donde se maximiza $F1$, los algoritmos tienden a detectar solamente las clases mayoritarias. Se muestra que eligiendo un punto de trabajo donde se maximiza la relación entre las sumas ponderadas $\hat{F}1$ y $\hat{ER}$, se detectan mejor las clases minoritarias sin afectar sensiblemente el desempeño global.

El segundo enfoque refiere a la utilización de redes neuronales profundas (de varias capas). En primer lugar, se presentan diferentes arquitecturas que han sido utilizadas para la detección de eventos sonoros [19, 35, 44, 49] y se entrenan con los datos de URBAN-SED. Se muestra que se obtienen mejores resultados cuando se utilizan redes convolucionales, ya que estas aprovechan la estructura de tiempo-frecuencia de las señales de audio. Luego, se presenta un enfoque novedoso para la detección de eventos sonoros urbanos utilizando una red *punta–a–punta* cuya entrada es la señal de audio y la salida es el vector de clasificación. Esta red se diseña en dos etapas: una para la extracción de características y otra para la clasificación. Para la etapa de extracción de características se propone un modelo (SMel) que busca aproximar la representación de la señal de audio como el espectrograma en bandas de frecuencia mel y que se basa en un banco de filtros. Con SMel se obtienen mejores resultados que con el modelo MST [65] recientemente publicado. Se realizan experimentos que muestran la utilidad de este tipo de enfoques para aumentar la capacidad de interpretación de los parámetros de la red después del entrenamiento. En particular, se estudia cómo cambian los filtros de la primera capa luego del entrenamiento, y se concluye que se puede diseñar un modelo con menos parámetros y que obtiene resultados similares.

Los resultados presentados durante todo el documento muestran que los algoritmos, principalmente los de aprendizaje profundo, tienden a detectar solo las clases mayoritarias. Esto muestra que puede ser necesario introducir modificaciones en las medidas de desempeño para priorizar la detección de eventos de las clases minoritarias. Además, se pueden realizar mejoras a los datos disponibles o diseñar sistemas más adecuados para esa tarea. A continuación, se proponen diferentes lineas de trabajo futuro para mejorar el desempeño de esos sistemas.

7.2. Trabajo futuro

Como trabajo futuro se propone completar la base de datos SonidosMVD con miras a mejorar la representación de las clases minoritarias. Además, se propone publicar esta nueva versión de la base de datos para que pueda ser utilizada en otros trabajos. Para mejorar el desempeño en las clases minoritarias, también puede ser

útil generar más datos sintetizados con las técnicas de *data augmentation*.

Se propone profundizar en el estudio de las medidas de desempeño para atacar el problema del desequilibrio de clases. Esto implica hacer un análisis exhaustivo de las medidas de desempeño que se han propuesto para este tipo de problemas y evaluarlas en esta aplicación en particular. Además, es interesante cambiar las estrategias de entrenamiento de las redes neuronales para lidiar con el problema del desequilibro de clases como se puede ver en [87, 88].

Para mejorar el desempeño de los algoritmos, se propone profundizar en el diseño de características específicas que extraigan información relevante para la detección de las clases de interés. En el Apéndice A se muestran algunos experimentos en este sentido, pero se debe profundizar en este tema para obtener mejores resultados.

Otro acercamiento interesante al problema de detección de eventos sonoros es utilizar redes neuronales basadas en regiones, como la presentada en [89]. Este enfoque se basa en Faster–RCNN, un modelo convolucional–recurrente para la detección de objetos en imágenes. Esta arquitectura incluye tres redes neuronales: una red para la extracción de características; una red donde se proponen regiones donde puede haber eventos sonoros; y una red encargada de clasificar dichas regiones.

Por otro lado, para mejorar los resultados de la clasificación, puede ser interesante un enfoque de procesamiento multimodal utilizando las grabaciones de audio y video conjuntamente. Un ejemplo de este tipo de enfoque es la arquitectura SoundNet [90] que incluye un clasificador de audio que es entrenado con información de video no etiquetado. Con esta arquitectura se aprende una representación de las señales de audio que se utiliza para entrenar un clasificador SVM y que obtiene mejores resultados que modelos que solo utilizan información del audio. Por otro lado, se pueden utilizar técnicas de detección de objetos en imágenes para clasificar los vehículos y mejorar la detección de las clases minoritarias (por ejemplo Motorcyle). Además, mientras que en el dominio del audio, la simultaneidad de fuentes sonoras puede dificultar la detección, en la imagen se podrían detectar *todos* los vehículos en simultáneo mientras que no hayan oclusiones. En contrapartida, este enfoque no es útil para detectar las clases del nivel de componentes (y por lo tanto tampoco del subordinado) ya que estos no son visibles en la escena.

Apéndice A

Clasificación de registros sonoros

En este apéndice, se presentan técnicas clasificación de registros sonoros urbanos. Se muestran resultados de sistemas de aprendizaje poco profundo entrenados con la base de datos *UrbanSound8k* (ver Capítulo 2). Además, se proponen nuevas características diseñadas para mejorar la detección de algunas clases de interés.

A.1. Baseline

Salamon y Bello [17] proponen un sistema de base (*baseline*) para clasificar las fuentes sonoras de *UrbanSound8k*. En dicho sistema, se extraen características en ventanas de 23.2 ms solapadas un 50 %. Para cada *frame*, se calcula la energía en 40 bandas mel (entre 0 y 22050 Hz) y luego se extraen 25 coeficientes MFCC (ver Capítulo 4). Estas características se integran temporalmente para extraer la siguiente información estadística de cada componente: mínimo, máximo, mediana, media, varianza, oblicuidad, kurtosis y la media y varianza de la primera y segunda derivada.

En [17] se presentan pruebas con 4 clasificadores: árboles de decisión (J48), 5–NN, Random Forest (500 árboles) y SVM (kernel RBF). De los experimentos resulta que los mejores desempeños se tienen con Random Forest, por lo que se decide trabajar con dicho clasificador .

Como primer aproximación al problema, se replica el sistema descrito anteriormente, utilizando la librería *librosa* para extraer las características y *scikit–learn* para entrenar y clasificar los datos. El porcentaje de acierto es de 68.7 % y en la Figura A.1 se ilustra la matriz de confusión, donde se pueden ver los aciertos y errores para cada una de las clases.

A.2. Extracción de nuevas características

Como se describe en el Capítulo 4, las características calculadas en el *baseline*, son genéricas y originalmente fueron concebidos para describir el contenido espectral de señales de voz. En este apartado se describe un método para extraer características específicas para la clasificación de sonidos urbanos y evaluar su

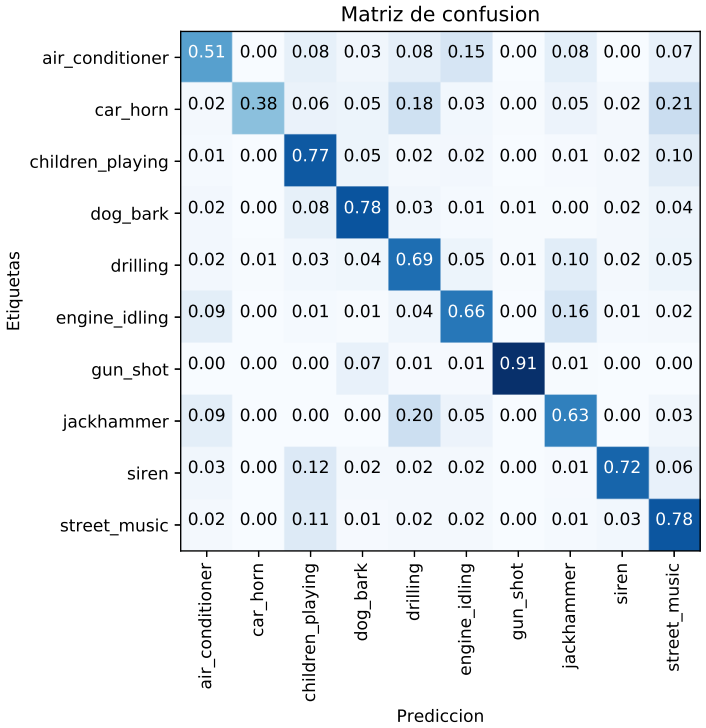


Figura A.1: Matriz de confusión del sistema baseline

desempeño. Para realizar esta evaluación se agregan dichas características a las calculadas en el *baseline* y se comparan los resultados obtenidos. De esta forma, se espera mejorar los resultados del *baseline* sumando conocimiento a priori sobre las clases de sonidos estudiados. En particular, se busca diseñar características que describan correctamente la clase SIREN ya que tiene un porcentaje de acierto de 72 % y una alta confusión con clases en principio muy diferentes como CHILDREN PLAYING y STREET MUSIC (ver Figura A.1).

A.2.1. Método propuesto

Se propone un método basado en programación dinámica para encontrar funciones continuas en un mapa de tiempo–frecuencia. Si se tiene un mapa de tiempo–frecuencia $S(f,t)$ donde t representa el tiempo y f la frecuencia, se quiere encontrar una función continua $y(t)$, con derivada acotada, $\left|\frac{dy(t)}{dt}\right| < C$, de forma que describa el camino filiforme de una fuente sonora monotónica. En ese caso, $S\left(y(t),t\right)$, representa la potencia instantánea de dicha fuente. En lo que sigue se utiliza la notación $S[i,j]$ y $y[j]$ para especificar que se trabaja en tiempo discreto.

En la Figura A.2 se puede ver el diagrama de bloques del método propuesto. Este proceso se realiza en dos etapas. En primer lugar, se calcula el espectrograma y se pre–procesa ponderando con una curva que simula la respuesta en frecuencia

del sistema auditivo. Además, se utiliza un filtro de mediana para eliminar componentes ruidosas. Luego, se utiliza un algoritmo de programación dinámica para encontrar la función de ganancia $g[i, j]$ y el camino óptimo $y[j]$. Por último, se calculan 10 características extraídas de $g[i, j]$ e $y[j]$.

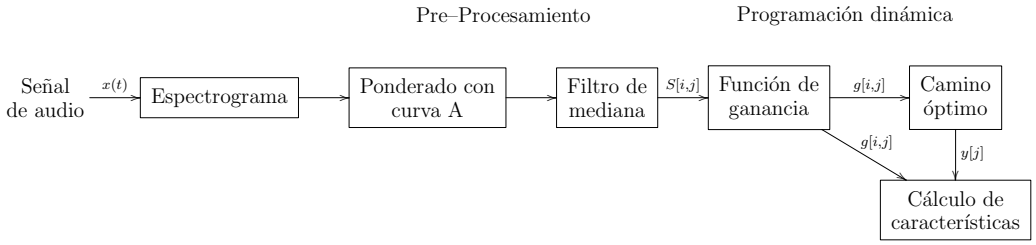


Figura A.2: Diagrama de bloques.

Pre–procesamiento

El espectrograma se calcula de la misma forma que en el *baseline*, con ventana de Hann de 1024 muestras y 50 % de solapamiento. Trabajamos con archivos con una frecuencia de muestreo de $f_s = 44100Hz$, por lo que la ventana es de $23.2ms$.

El objetivo del pre–procesamiento es limpiar el espectrograma de forma de simplificar el proceso de búsqueda de la función continua. Para ello, el primer paso es ponderar la energía en cada bin de frecuencia del espectrograma según la curva de ponderación A. Este proceso se realiza con la implementación disponible en la librería *librosa* [61]. En la Figura A.3 puede verse dicha curva extraída de la documentación de la propia librería .

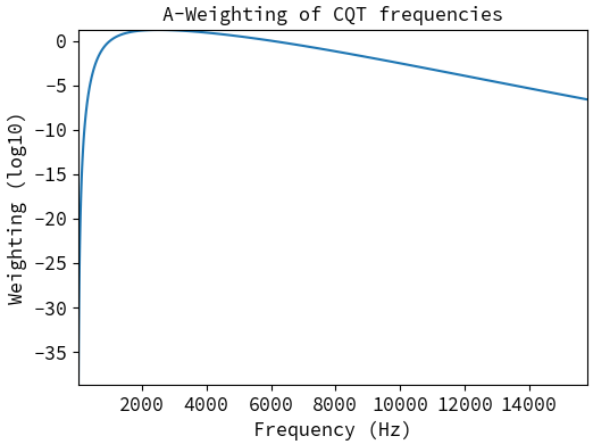
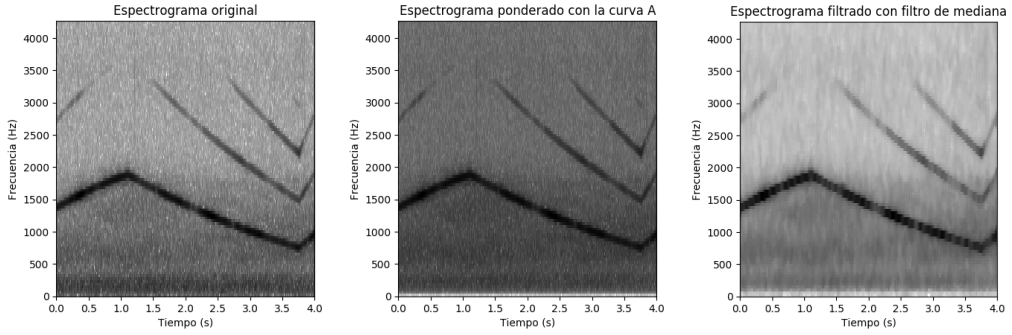


Figura A.3: Curva de ponderación A extraída de documentación de *librosa* [61].

Luego, se aplica un filtro de mediana en dos dimensiones con núcleo de tamaño 5x5 con el objetivo de atenuar las componentes ruidosas. En la Figura A.4 se puede ver un ejemplo de espectrograma de sirena y los resultados de los dos pasos del preprocesamiento.



(a) Original. (b) Ponderado. (c) Filtrado.

Figura A.4: Ejemplo del proceso de pre–procesamiento.

Algoritmo de Programación dinámica

Para encontrar el camino en el espectrograma $S[i, j]$ que represente una señal filiforme se utiliza un algoritmo de programación dinámica. La variable i representa cada bin de frecuencia y j cada salto en el tiempo, por lo tanto:

$$i = 0 \dots N_i = N/2 + 1 \tag{A.1}$$
$$j = 0 \dots N_j = \lfloor N_T / N_{hop} \rfloor, \tag{A.2}$$

donde N es el largo de la ventana, N_T es el número de muestras de la señal y $N_{hop} = N/2$ es el número de muestras del salto en el tiempo. En cada punto (i, j) se calcula la función de ganancia dada por:

$$\begin{aligned} g[i,j] = \text{máx}\{ & g[i-2, j-1] + 0.7 \times S[i,j], \\ & g[i-1, j-1] + 0.9 \times S[i,j], \\ & g[i, j-1] + S[i,j], \\ & g[i+1, j-1] + 0.9 \times S[i,j], \\ & g[i+2, j-1] + 0.7 \times S[i,j]\} \end{aligned}$$

De esta forma, se busca darle más peso al bin de frecuencia i, un poco menos a los bins $i \pm 1$ y menos aún a los bins $i \pm 2$. Por lo tanto, el módulo de la derivada de la función objetivo estará acotado por:

$$\left| \frac{dy(t)}{dt} \right| \leq \frac{2\Delta f}{\Delta T}, \tag{A.3}$$

donde $\Delta f = f_s/N$ es la resolución en frecuencia y $\Delta T = N_{hop}/f_s$ es el tiempo de hop. Entonces se tiene que:

$$\left|\frac{dy(t)}{dt}\right| \leq \frac{2f_s/N}{N_{hop}/f_s} = \frac{2f_s^2}{N \times N_{hop}}. \tag{A.4}$$

Cuando se calcula la función de ganancia, se guardan las decisiones que se toman en cada paso. Luego de calcular $g[i,j]$ se encuentra el argumento que maximiza $g[i,N_j]$ y se obtiene el camino óptimo $y[j]$ recurriendo a las decisiones hacia atrás. En la Figura A.5 se puede ver el camino óptimo $y[j]$ encontrado para un ejemplo y la función de ganancia en el último instante $g[i,N_j]$. Se puede ver que el camino se construye desde el punto que maximiza a $g[i,N_j]$, aproximadamente en 1000 Hz. Como la función de ganancia es acumulativa, es necesario normalizarla por la energía total de la señal y el número de pasos en el tiempo de forma de independizar la medida del largo de la señal y su energía.

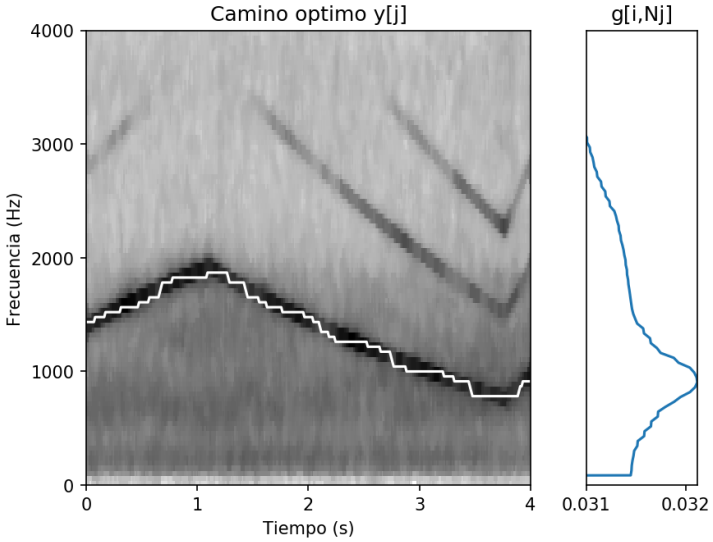


Figura A.5: Camino óptimo para un ejemplo de sirena.

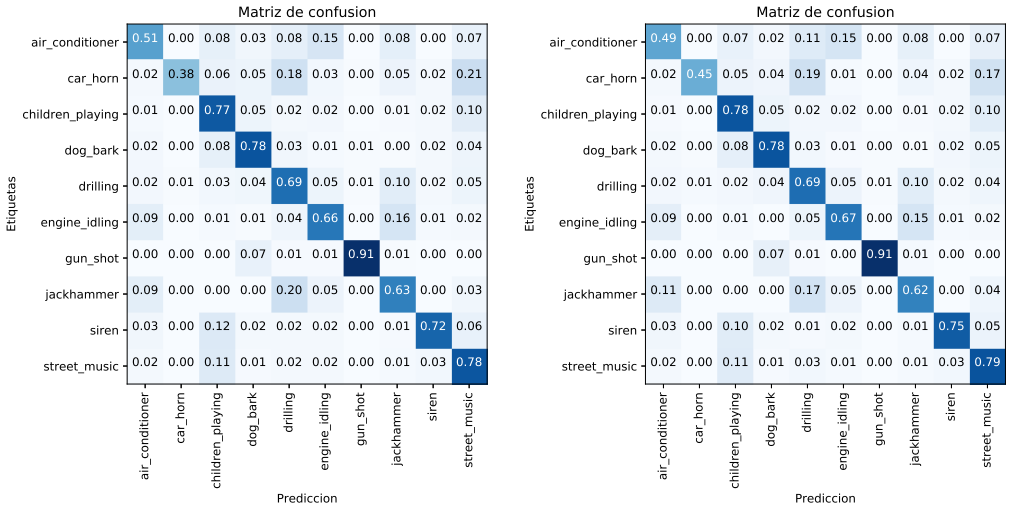
Por último se extraen las siguientes características:

- Máximo de $g[i,N_j]$ y el argumento i que lo maximiza. Si el máximo valor de la función de ganancia en el último instante es alto indica presencia de una función continua. El argumento puede ayudar en la discriminación entre sirenas y otras funciones continuas en otras bandas de frecuencias.
- Mínimo, máximo, media y varianza del camino $y[j]$. Media y varianza de las derivadas de primer y segundo orden de $y[j]$. Estas últimas características buscan representar la forma del camino $y[j]$.

En total, se suman 10 características a las 275 que se incluían en el baseline.

A.3. Experimentos

Para realizar los experimentos se se utiliza el mismo algoritmo que el presentado en el baseline [17], Random Forest con 500 árboles. Luego, se implementa la extracción de las nuevas características propuestas y se suman a las 275 calculadas en el baseline. En las Figuras A.6a y A.6b se ilustran las matrices de confusión del baseline y el sistema que utiliza las nuevas características respectivamente. En la nueva matriz de confusión se puede ver que en la clase Siren se mejora la detección de 72 % a 75 %. Además el desempeño total del sistema mejora de 68.7 % a 69.1 %.



(a) Baseline.

(b) Baseline y nuevas características.

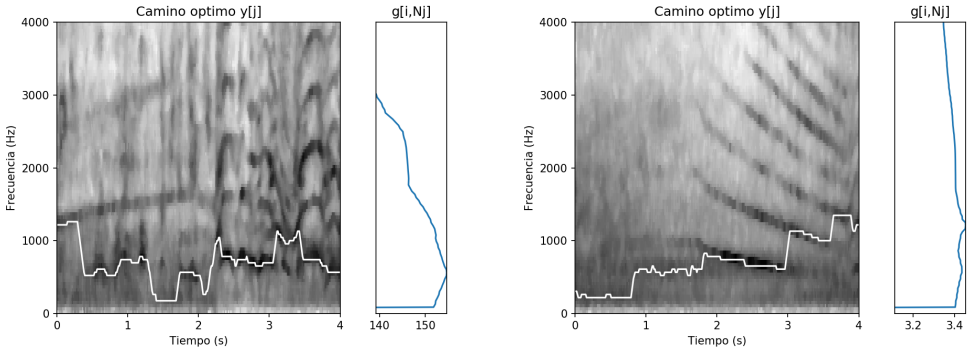
Figura A.6: Matrices de confusión del baseline y de las nuevas características.

A.4. Casos de error

Los casos de error se dividen en dos tipos, falsos negativos y falsos positivos. Los primeros son aquellos casos en que la etiqueta es Siren pero el algoritmo detecta otra clase. Los segundos son casos donde la etiqueta NO es Siren, pero el algoritmo los detecta como tal.

En la Figura A.7 se pueden ver dos casos de falsos negativos. En el primero se tiene que la sirena se escucha lejos en la grabación y se escucha una conversación por lo que se termina confundiendo con Children playing. En el segundo caso se tiene una sirena se va acercando en el transcurso de la grabación, por lo que al principio su potencia es muy baja y cuando se acerca mucho se generan armónicos por distorsión. En este caso se confunde con Street Music.

Otros casos especiales de falsos negativos los que corresponden a otros tipos de sirenas. Un ejemplo de esto, es la sirena de bombero como el de la Figura A.8. También hay sonidos de alarmas de auto etiquetados como sirena. Está claro que



(a) Siren que se detecta como Children playing.

(b) Siren que se detecta como Street Music.

Figura A.7: Ejemplos de Falsos Negativos.

este tipo de sonidos no tiene una estructura filiforme en el espectrograma, por lo que no cumple con las hipótesis que se plantea.

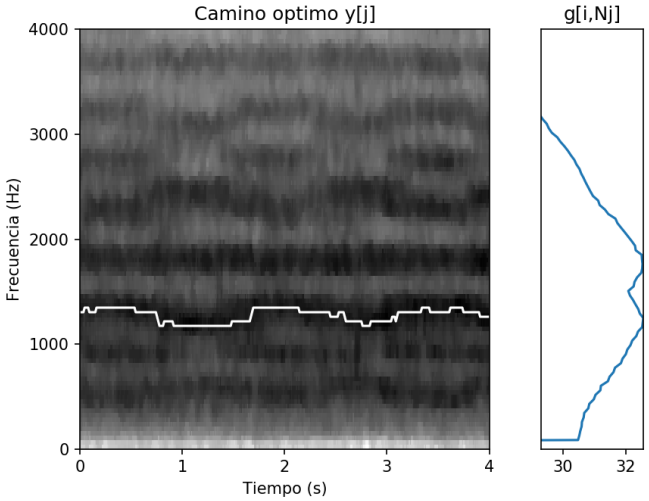
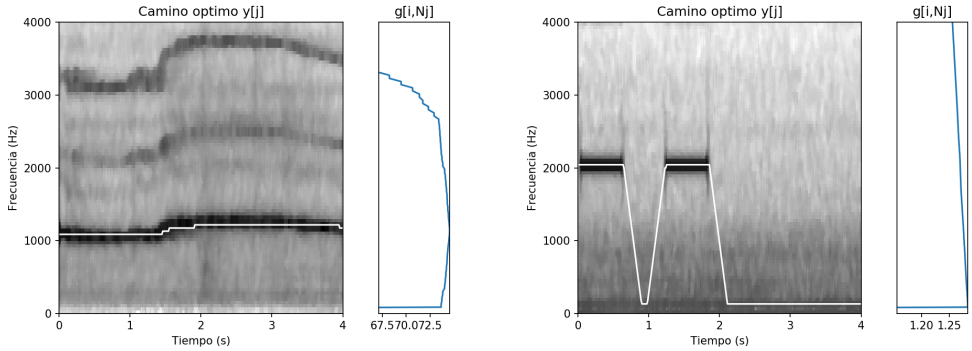


Figura A.8: Ejemplo de sirena de bombero que no es correctamente detectada.

En la Figura A.9 se pueden ver dos casos de falsos positivos. En el primero se tiene un caso de Street Music que es una grabación de una flauta cuyo espectro resulta muy similar al de las sirenas. En el segundo se tiene un aire acondicionado que genera un tono de frecuencia constante de mucha potencia y a una frecuencia similar a las sirenas.



(a) Street Music que se detecta como Siren.

(b) Air conditioner que se detecta como Siren.

Figura A.9: Ejemplos de Falsos Positivos.

A.5. Discusión

En este apéndice se presenta un algoritmo basado en programación dinámica para detectar la presencia de fuentes sonoras filiformes en mapas de tiempo–frecuencia. Del resultado de dicho algoritmo se extraen 10 características con el objetivo de mejorar la detección de la clase Siren en la base de datos *UrbanSound8k*.

Agregando las características propuestas al *baseline*, se mejora la detección de sirena de 72 % a 75 % (de 763 a 799 casos correctamente detectados). Además, el resultado total del sistema mejora de 68.7 % a 69.1 %. Por lo tanto, a pesar de que los MFCC son características que demostraron funcionar bien en muchos problemas, sumar información a priori de las clases estudiadas, puede ayudar a mejorar la clasificación.

A partir de los casos de error se concluye que el algoritmo no da buenos resultados cuando la sirena no es la única fuente y/o la energía de dicha fuente es muy baja con respecto al total de la señal. También resulta que fuentes similares como instrumentos de viento o equipos electrónicos que generan tonos a frecuencias constantes pueden confundirse con sirenas.

Referencias

[1] R. Murray Schafer. *The soundscape: our sonic enviroment and the tuning of the world*. Destiny Books, 1994.

[2] Alice Elizabeth González. *Contaminación sonora y Derechos Humanos*. Defensoría del vecino, 2012.

[3] Dirkjan Krijnders Daniel Steele y Catherine Guastavino. The sensor city initiative: cognitive sensors for soundscape transformations. En *Geoinformatics for City Transformations*, páginas 243–253. Technical University of Ostrava, January 2013.

[4] Defensoría del vecino. *Noveno Informe Anual*. Defensoría del vecino, 2015.

[5] Defensoría del vecino. *Décimo Informe Anual*. Defensoría del vecino, 2016.

[6] Defensoría del vecino. *Undécimo Informe Anual*. Defensoría del vecino, 2017.

[7] Simon Fraser University. The world soundscape project. `www.sfu.ca/~truax/wsp.html`. [Internet; descargado 20-08-2016].

[8] R. Murray Schafer. *The New Soundscape: A handbook for the Modern Music Teacher*. BMI Canada Limited, 1969.

[9] Daniel Maggiollo. Soundscape Research in Uruguay. *Soundscape: The Journal of Acoustic Ecology*, 4(1):51–52, 2003.

[10] Escuela Universitaria de Música. Página web del proyecto paisaje sonoro Uruguay. `www.eumus.edu.uy/eme/ps/`. [Internet; descargado 23-05-2016].

[11] IMFIA. Página web del Departamento de Ingeniería Ambiental del IMFIA. `www.fing.edu.uy/imfia/?q=node/229`. [Internet; descargado 23-05-2016].

[12] Alice Elizabeth González et. al. Mapa acústico de Montevideo. informe final. Reporte técnico, Convenio IMFIA-IMM, 1999.

[13] Juan Pablo Bello, Charlie Mydlarz, y Justin Salamon. *Computational Analyis of Sound Scenes and Events*, chapter 13 Sound Analysis in Smart Cities. Springer, 2017.

[14] Melody Baglione Charlie Mydlarz, Charles Shamoon y Michael Pimpinella. The design and calibration of low cost urban acoustic sensing devices. En *EuroNoise 2015*, páginas 2345–2350. European Acoustics Association, 2015.

[15] Charlie Mydlarz, Samuel Nacach, Agnieszka Roginska, Tae Hong Park, Eric Rosenthal, y Michelle Temple. The implementation of MEMS microphones for urban sound sensing. En *Audio Engineering Society Convention 137*, Oct 2014.

[16] Justin Salamon y Juan Pablo Bello. Unsupervised feature learning for urban sound classification. En *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, páginas 171–175, 2015.

[17] J. Salamon, C. Jacoby, y J. P. Bello. A dataset and taxonomy for urban sound research. En *22st ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, Nov. 2014.

[18] Justin Salamon y Juan Pablo Bello. Feature learning with deep scattering for urban sound analysis. En *2015 European Signal Processing Conference (EUSIPCO)*, páginas 724–728, 2015.

[19] J. Salamon, D. MacConnell, M. Cartwright, P. Li, y J. P. Bello. Scaper: A library for soundscape synthesis and augmentation. En *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, october 2017.

[20] Andrey Temko, Climent Nadeu, Dušan Macho, Robert Malkin, Christian Zieger, y Maurizio Omologo. *Acoustic Event Detection and Classification*, páginas 61–73. Springer London, London, 2009.

[21] Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, y Mark D. Plumbley. Detection and classification of acoustic scenes and events. *IEEE Transactions on Multimedia*, 17:1733 – 1746, Oct. 2015.

[22] Annamaria Mesaros, Toni Heittola, y Tuomas Virtanen. TUT database for acoustic scene classification and sound event detection. En *In 24rd European Signal Processing Conference 2016 (EUSIPCO 2016), Budapest, Hungary*, 2016.

[23] A. Mesaros, T. Heittola, y T. Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162, 2016.

[24] Tuomas Virtanen, Mark D. Plumbley, y Dan Ellis. *Computational Analyis of Sound Scenes and Events*, chapter 1 Introduction to Sound Scene and Event Analysis. Springer, 2017.

[25] Catherine Guastavino. *Computational Analyis of Sound Scenes and Events*, chapter 7 Everyday Sound Categorization. Springer, 2017.

[26] Annamaria Mesaros, Toni Heittola, y Dan Ellis. *Computational Analyis of Sound Scenes and Events*, chapter 6 Datasets and Evaluation. Springer, 2017.

[27] New York University. Página web del proyecto SONYC. `wp.nyu.edu/sonyc/`. [Internet; descargado 22-09-2016].

[28] Karol J. Piczak. ESC: Dataset for environmental sound classification. En *23rd ACM international conference*, Oct. 2015.

[29] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, y Marvin Ritter. Audio Set: An ontology and human-labeled dataset for audio events. En *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[30] Annamaria Mesaros y Toni Heittola. Recording and annotation procedure. `www.cs.tut.fi/sgn/arg/dcase2016/task-sound-event-detection-in-real-life-audio#audio-dataset`. [Internet; descargado 17-12-2018].

[31] Max Planck Institute for Psycholinguistics. ELAN - the lenguage active. `tla.mpi.nl/tools/tla-tools/elan/`. [Internet; descargado 17-12-2018].

[32] Ian H. Witten y Eibe Frank. *Data Mining. Practical Machine Learning Tools and Techniques.*, chapter 5 Credibility: Evaluating What's Been Learned, páginas 143–185. Elseiver, 2005.

[33] Troy Raeder, George Forman, y Nitesh V. Chawla. *Data Mining: Foundations and Intelligent Paradigms.*, volume Volume 1: Clustering, Association and Classification, chapter 12 Learning from Imbalanced Data: Evaluation Matters, page 315–331. Springer, 2012.

[34] George Forman y Martin Scholz. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explor. Newsl.*, 12(1):49–57, November 2010.

[35] Annamaria Mesaros, Toni Heittola, Aleksandr Diment, Benjamin Elizalde, Ankit Shah, Emmanuel Vincent, Bhiksha Raj, y Tuomas Virtanen. DCASE 2017 challenge setup: Tasks, datasets and baseline system. Reporte técnico, DCASE2017 Challenge, September 2017.

[36] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, y M. D. Plumbley. Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(2):379–393, Feb 2018.

[37] Sacha Krstulović. *Computational Analyis of Sound Scenes and Events*, chapter 12 Audio Event Recognition in the Smart Home. Springer, 2017.

[38] R. Alejo, J. A. Antonio, R. M. Valdovinos, y J. H. Pacheco-Sánchez. Assessments metrics for multi-class imbalance learning: A preliminary study. En Jesús Ariel Carrasco-Ochoa, José Francisco Martínez-Trinidad, Joaquín Salas Rodríguez, y Gabriella Sanniti di Baja, editores, *Pattern Recognition*, páginas 335–343, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[39] Tampere University. Sound event detection in real life audio. task results. `www.cs.tut.fi/sgn/arg/dcase2016/task-results-sound-event-detection-in-real-life-audio`, 2016. [Internet; descargado 13-01-2019].

[40] Tampere University. Sound event detection in real life audio. challenge results. `www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-sound-event-detection-in-real-life-audio-results`, 2017. [Internet; descargado 13-01-2019].

[41] M Hossin y M.N. Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 5(2), March 2015.

[42] Matthias Zöhrer y Franz Pernkopf. Gated recurrent networks applied to acoustic scene classification and acoustic event detection. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[43] Sharath Adavanne y Tuomas Virtanen. A report on sound event detection with different binaural features. En *Detection and Classification of Acoustic Scenes and Events 2017*, 2017.

[44] Sharath Adavanne, Giambattista Parascandolo, Pasi Pertila, Toni Heittola, y Tuomas Virtanen. Sound event detection in multichannel audio using spatial and harmonic features. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[45] Toan H. Vu y Jia-Ching Wang. Acoustic scene and event recognition using recurrent neural networks. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[46] Arseniy Gorin, Nurtas Makhazhanov, y Nickolay Shmyrev. DCASE 2016 sound event detection system based on convolutional neural network. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[47] Jianchao Zhou. Sound event detection in multichannel audio LSTM network. Reporte técnico, DCASE2017 Challenge, September 2017.

[48] Il-Young Jeong, Subin Lee, Yoonchang Han, y Kyogu Lee. Audio event detection using multiple-input convolutional neural network. Reporte técnico, DCASE2017 Challenge, September 2017.

[49] Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, y Tuomas Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *Transactions on Audio, Speech and Language Processing: Special issue on Sound Scene and Event Analysis*, 25(6):1291–1303, June 2017.

[50] Youngmoo E. Kim y Brian Whitman. Singer identification in popular music recordings using voice coding features. En *Interational Conference on Music Information Retrieval*, page 164–169, 2002.

[51] Perfecto Herrera-Boyer, Anssi Klapuri, y Manuel Davy. chapter Automatic Classification of Pitched Musical Instrument Sounds, page 163–200. Springer, New York, 2006.

[52] Audio research group of Tampere University of Technology. DCASE 2016 Challenge. `www.cs.tut.fi/sgn/arg/dcase2016/challenge`. [Internet; descargado 22-09-2016].

[53] Benjamin Elizalde, Anurag Kumar, Ankit Shah, Rohan Badlani, Emmanuel Vincent, Bhiksha Raj, y Ian Lane. Experimentation on the dcase challenge 2016: Task 1 - acoustic scene classification and task 3 - sound event detection in real life audio. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[54] Qiuqiang Kong, Iwnoa Sobieraj, Wenwu Wang, y Mark Plumbley. Deep neural network baseline for dcase challenge 2016. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[55] Dai Wei, Juncheng Li, Phuong Pham, Samarjit Das, Shuhui Qu, y Florian Metze. Sound Event Detection for Real Life Audio DCASE Challenge. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[56] Chun-Hao Wang, Jun-Kai You, y Yi-Wen Liu. Sound event detection from real-life audio by training a long short-term memory network with mono and stereo features. Reporte técnico, DCASE2017 Challenge, September 2017.

[57] Rui Lu y Zhiyao Duan. Bidirectional GRU for sound event detection. Reporte técnico, DCASE2017 Challenge, September 2017.

[58] S.B. Davis y P. Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[59] Yuxuan Wang, Pascal Getreuer, Thad Hughes, Richard F. Lyon, y Rif A. Saurous. Trainable frontend for robust and far-field keyword spotting. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017.

[60] V. Lostanlen, J. Salamon, M. Cartwright, B. McFee, A. Farnsworth, S. Kelling, y J. P. Bello. Per-Channel Energy Normalization: Why and how. *IEEE Signal Processing Letters*, 26(1):39–43, Jan. 2019.

[61] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Eric Battenberg, Josh Moore, Dan Ellis, Ryuichi Yamamoto, Rachel Bittner, Douglas Repetto, Petr Viktorin, João Felipe Santos, y Adrian Holovaty. librosa: 0.4.1, October 2015.

[62] Alex Krizhevsky, Ilya Sutskever, y Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. En F. Pereira, C. J. C. Burges, L. Bottou, y K. Q. Weinberger, editores, *Advances in Neural Information Processing Systems 25*, páginas 1097–1105. Curran Associates, Inc., 2012.

[63] John Thickstun, Zaid Harchaoui, y Sham M. Kakade. Learning features of music from scratch. En *5th International Conference on Learning Representations - ICLR 2017*, 2017.

[64] Jongpil Lee, Jiyoung Park, Keunhyoung L. Kim, y Juhan Nam. SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification. *Applied Sciences*, 8(1), 2018.

[65] Tycho Max Sylvester Tax, Jose Luis Diez Antich, Hendrik Purwins, y Lars Maaløe. Utilizing domain knowledge in end-to-end audio processing. En *31st Conference on Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, 2017.

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, y E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[67] S. Nesmachnow. Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. *Revista de la Asociación de Ingenieros del Uruguay*, 61:12–15, 2010.

[68] Brian McFee. *Computational Analyis of Sound Scenes and Events*, chapter 5 Statistical Methods for Scene and Event Classification. Springer, 2017.

[69] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, y Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[70] Ian Goodfellow, Yoshua Bengio, y Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[71] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, páginas 65–386, 1958.

[72] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[73] John Duchi, Elad Hazan, y Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[74] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[75] Diederik P. Kingma y Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[76] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, y Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. En *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1724–1734, Doha, Qatar, October 2014.

[77] Sepp Hochreiter y Jürgen Schmidhuber. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Neural Computation*, 9:1735–1780, 1997.

[78] François Chollet et al. Keras. `https://keras.io`, 2015.

[79] K. Simonyan y A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[80] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, y Andrew Rabinovich. Going deeper with convolutions. Oct 2015.

[81] Zoltán Tüske, Pavel Golik, Ralf Schlüter, y Hermann Ney. Acoustic modeling with deep neural networks using raw time signal for lvcsr. En *INTERSPEECH*, 2014.

[82] Neil Zeghidour, Nicolas Usunier, Gabriel Synnaeve, Ronan Collobert, y Emmanuel Dupoux. End-to-end speech recognition from the raw waveform. En *Interspeech*, 2018.

[83] Tara N. Sainath, Ron J. Weiss, Kevin W. Wilson, Arun Narayanan, Michiel Bacchiani, Bo Li, Ehsan Variani, Izhak Shafran, Andrew Senior, Kean Chin, Ananya Misra, y Chanwoo Kim. *Raw Multichannel Processing Using Deep Neural Networks*. 2017.

[84] Giacomo Valenti, Adrien Daniel, y Nicholas Evans. End-to-end automatic speaker verification with evolving recurrent neural networks. 2018.

[85] Sander Dieleman y Benjamin Schrauwen. End-to-end learning for music audio. En *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, páginas 6964–6968, 05 2014.

[86] Xinqi Zhu y Michael Bain. B-CNN: branch convolutional neural network for hierarchical classification. 2017.

[87] Qi Dong, Shaogang Gong, y Xiatian Zhu. Imbalanced deep learning by minority class incremental rectification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 05 2018.

[88] Mateusz Buda, Atsuto Maki, y Maciej A. Mazurowski. A systematic study of the classimbalance problem in convolutional neural networks. *Neural Networks*, 106:249 – 259, 2018.

[89] Chieh-Chi Kao, Weiran Wang, Ming Sun, y Chao Wang. R-CRNN: Region-based convolutional recurrent neural network for audio event detection. En *Interspeech 2018*, Sep. 2018.

[90] Yusuf Aytar, Carl Vondrick, y Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. En *Advances in Neural Information Processing Systems*, 2016.

Índice de tablas

Índice de figuras

Esta es la última página.
Compilado el lunes 13 mayo, 2019.
http://iie.fing.edu.uy/