UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Detección de voz y música en un corpus a gran escala de eventos de audio

Máster Universitario en
Investigación e Innovación en TIC (i^2 TIC)

Autor: de Benito Gorrón, Diego
Tutor: González Rodríguez, Joaquín

Fecha: Junio 2018

Trabajo Fin de Máster

Detección de voz y música en un corpus a gran escala de eventos de audio

Autor: Diego de Benito Gorrón
Tutor: Joaquín González Rodríguez

Máster Universitario en Investigación e Innovación en TIC (i^2TIC)
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Junio 2018

AUDIAS - Audio, Data Intelligence and Speech
http://audias.ii.uam.es/

Resumen

El **reconocimiento de eventos acústicos** es la capacidad para **extraer información** de un suceso a partir de los **sonidos** producidos por el mismo. El desarrollo de esta habilidad, básica en el sistema auditivo humano, **en la inteligencia computacional** es un problema que se está abordando mediante la investigación en modelos de aprendizaje automático (*machine learning*) como las **redes neuronales**.

En este **Trabajo Fin de Máster** se estudia la aplicación de diferentes arquitecturas de redes neuronales a la **detección de eventos de voz y de música** sobre un conjunto de **77.396 segmentos** de audio de 10 segundos (**216 horas**) obtenidos de la base de datos **Google AudioSet**. Estos segmentos pertenecen a fragmentos de **vídeos de la plataforma YouTube**.

Se proponen y comparan dos enfoques distintos para el problema: uno de ellos es el entrenamiento de **dos redes neuronales separadas**, una para detección de presencia de voz y otra para detección de presencia de música, y el otro consiste en el entrenamiento de **una red neuronal conjunta** que se enfrente simultáneamente a ambas tareas. Entre las arquitecturas estudiadas se encuentran las **redes *fully-connected***, las **redes convolucionales** y las **redes LSTM** (*Long Short-Term Memory*).

A lo largo del trabajo se describe la **organización de la base de datos**, la **construcción de los conjuntos de datos** empleados y el **diseño de los modelos** propuestos. Además, se proveen **resultados comparativos** de las distintas configuraciones evaluadas, tanto en **rendimiento** como en **complejidad** de los modelos.

Palabras clave

aprendizaje automático, *deep learning*, redes neuronales, redes convolucionales, LSTM, audio, voz, música, reconocimiento de eventos de audio

Abstract

Acoustic event recognition is the ability to **extract information** about ocurrences from the sounds produced by them. Such is a basic skill for the human auditory system, but its development in **computational intelligence** is currently an active research field involving machine learning models such as **neural networks**.

This **Master's Thesis** aims to study the implementation of several neural network architectures to **speech and music event recognition** over a collection of **77,936** ten seconds **audio segments** (**216 hours**), obtained from the **Google AudioSet** dataset. These segments belong to **YouTube videos**.

Two different approaches are proposed and compared: one of them is training **two separated neural networks** for **speech event detection** and **music event detection**, while the other one consists on training a **joint neural network** to tackle **both tasks** at the same time. Among the studied architectures, **fully-connected networks**, **convolutional networks** and **LSTM** (Long Short-Term Memory) are included.

Along this report, the **organization of the dataset**, the **selection of the used segments** and the **design of the proposed models** are described. Additionally, **comparative results** of the evaluated settings are provided in terms of **performance** and **model complexity**.

Keywords

machine learning, deep learning, neural networks, convolutional networks, LSTM, audio, speech, music, audio event recognition

Agradecimientos

A mis familiares, amigas y amigos.

A mi tutor, Joaquín González, por ofrecerme el proyecto y todas las facilidades para llevarlo a cabo, y a Doroteo Torre por la inestimable labor de obtención de la base de datos AudioSet.

A Alicia Lozano y Rubén Zazo, por su valiosa ayuda y sus consejos sobre redes neuronales, y al resto de integrantes de AUDIAS.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

La **voz humana** y la **música** son dos de las diversas clases de contenido que pueden ser encontradas en las **señales de audio**. Inferir la presencia de estos contenidos a partir de las características de la señal es el objetivo de campos de investigación como la **segmentación de audio** [2] o la **detección de actividad de voz** [3], soliendo proponerse estas tareas como paso previo a un análisis dirigido específicamente a voz o a música: por ejemplo, reconocimiento de locutor en el caso de señales de voz, o detección de tonalidad en el caso de señales musicales [4].

La **segmentación automática de audio** es una de las líneas de investigación del grupo AUDIAS [1]. Este campo, también denominado "transcripción rica de audio" (*rich transcription*) [5], está enfocado hacia la **obtención automática de descripciones del contenido** de las señales de audio. Dichas señales pueden proceder de distintas fuentes y haber sido obtenidas bajo diversas condiciones acústicas, dando lugar a una **variabilidad** que los modelos de transcripción propuestos deben afrontar.

El desarrollo de grandes colecciones de eventos de audio etiquetados, como el reciente **AudioSet de Google** [1], supone una gran oportunidad para la investigación en este campo mediante el desarrollo de modelos de clasificación basados en técnicas de aprendizaje automático [6–9].

El principal objetivo del presente Trabajo Fin de Máster es la **investigación y evaluación de modelos de aprendizaje automático** y *deep learning* aplicados a la **detección de eventos de audio** en la base de datos Google AudioSet, que cuenta con **5.000 horas de audio** etiquetadas manualmente en segmentos de 10 segundos. A diferencia de otros retos, como la competición DCASE [10], el trabajo se centrará en la identificación de dos de las etiquetas contenidas en **AudioSet**,

[1] http://audias.ii.uam.es/

las correspondientes a **música** y a **voz humana**.

El **máster universitario *i*^{2}TIC** (Investigación e Innovación en Tecnologías de la Información y la Comunicación) de la Escuela Politécnica Superior, en su itinerario de Inteligencia Computacional, supone un marco académico muy adecuado para el desarrollo de este Trabajo Fin de Máster, que guarda relación con asignaturas como Cálculo Intensivo y Manejo de Datos a Gran Escala, Aprendizaje Automático o Tratamiento de Información Temporal.

1.2. Objetivos

Los **objetivos concretos** propuestos para este trabajo son:

- Estudio, comprensión y descripción de la base de datos AudioSet de Google.
- Consulta del estado del arte del área, experimentos relacionados y resultados previos.
- Diseño de modelos de clasificación basados en diferentes arquitecturas de redes neuronales.
- Diseño y ejecución de experimentos para la evaluación del rendimiento de los modelos de clasificación propuestos.
- Interpretación y discusión de los resultados obtenidos.
- Generación de código orientado al manejo de la base de datos AudioSet y su uso en futuras investigaciones.

1.3. Organización de la memoria

La memoria del presente Trabajo Fin de Máster se divide en **cinco bloques o capítulos**:

- **Capítulo 1. Introducción.**
  Este capítulo plantea brevemente el **contexto** del trabajo en las líneas de investigación del grupo AUDIAS y la relación de las tareas llevadas a cabo con el itinerario de Inteligencia Computacional del máster i^2TIC. Se describen, además, los **objetivos** concretos del trabajo.
- **Capítulo 2. Estado del arte.**
  El objetivo de este capítulo es explicar los **conceptos**, las **técnicas** y la **terminología** que se utilizan en este trabajo o son de relevancia para el

desarrollo del mismo. En concreto, se abordan temas relacionados con el **procesamiento de audio digital** y con el **aprendizaje automático**.

- **Capítulo 3. Descripción y diseño de los experimentos.**
  En este capítulo se describe, en primer lugar, la **base de datos** utilizada (Google AudioSet), cómo está organizada y qué datos y etiquetas se han seleccionado. También se detalla la **obtención de características** a partir de los ficheros de audio y, además, se concretan los procesos de **construcción, entrenamiento y selección de los modelos** de clasificación propuestos.

- **Capítulo 4. Resultados.**
  Los **resultados obtenidos** por los distintos modelos propuestos se muestran en este capítulo de forma numérica y gráfica, acompañados por una **discusión e interpretación** de los mismos.

- **Capítulo 5. Conclusiones y trabajo futuro.**
  El capítulo final recoge las **conclusiones** obtenidas tras el desarrollo de este trabajo y la **evaluación** del cumplimiento de los objetivos propuestos, y describe qué tareas podrían dar **continuidad** a la investigación que se ha llevado a cabo.

Capítulo 2

Estado del arte

2.1. Detección de voz y música: un enfoque clásico

El **procesamiento de señales de audio** ha sido aplicado a la obtención automatizada de información de alto nivel sobre el contenido de las mismas desde antes del auge de las técnicas de aprendizaje automático (*Machine Learning*).

Son muchas y muy diversas las tareas que han sido investigadas en este campo con propuestas alejadas de modelos como las redes neuronales. Esta familia de modelos puede ser denominada de forma general como **"enfoque clásico"**.

Los modelos clásicos son en muchas ocasiones soluciones ad-hoc, en las que las características extraídas de las señales son analizadas para establecer las **reglas de decisión óptimas**. También es frecuente el uso de **herramientas probabilísticas** como modelos de mezcla de Gaussianas (GMM, *Gaussian Mixture Models*), máquinas de vectores de soporte (SVM, *Support Vector Machines*) [11], *i-vectors* o modelos ocultos de Markov (HMM, *Hidden Markov Models*) [12].

2.1.1. Segmentación automática de audio

La **segmentación automática de audio** [2] consiste en la identificación de **tramos temporales** de una señal acústica cuyo contenido pertenezca a **clases diferenciadas**. Estas clases dependen de la aplicación: se pueden referir a distintos locutores, idiomas, instrumentos musicales, o simplemente a categorías generales como voz humana, música o ruido.

La **detección de actividad de voz** (VAD, *Voice Activity Detection*) es un caso concreto de segmentación automática de audio. Su objetivo es detectar aquellos **tramos de señal que contienen voz** [3]. Esta línea de investigación cuenta con evaluaciones a nivel internacional como **OpenSAT** [13], y debe enfrentarse

al **ruido** que puede aparecer en la señal debido a **problemas de adquisición**, a la presencia de **fuentes acústicas adicionales** o a los **efectos del entorno acústico** (por ejemplo, reverberación).

La **detección de actividad musical**, es decir, la **identificación de segmentos** de una señal acústica **en los que aparecen contenidos musicales**, es una tarea menos común, pero también de interés. En este escenario surgen problemáticas específicas, como la propia **definición de musicalidad** y la **gran variabilidad** que puede encontrarse en este tipo de señales. Estos obstáculos se añaden a los posibles problemas de adquisición, fuentes sonoras adicionales y entorno acústico.

En el Trabajo Fin de Grado "Detección de música en contenidos multimedia mediante ritmo y armonía" [14] se propone un **sistema de detección de actividad musical** en el que la presencia de contenidos musicales se aproxima mediante la **detección por separado de componentes rítmicas y componentes armónicas**. Este sistema es una **aproximación clásica** a la tarea de segmentación, y un ejemplo de cómo un modelo puede ser **diseñado de manera concreta** para resolver un problema dado.

La detección de actividad de voz y de música suelen ser tareas previas a la **resolución de problemas específicos** de las señales de voz o de música: en señales de voz, detección de idioma [15], de locutor [16] o transcripción de texto, y en el caso de señales musicales, detección de tonalidad [4], transcripción de melodía [17] o clasificación por género musical [18]. Adicionalmente, contar con información sobre la presencia de música puede ser de utilidad para un sistema de detección de actividad de voz, como se propone en [19].

2.1.2. Características de audio

Las **señales de audio digital** son, en esencia, **secuencias de muestras discretas** representadas digitalmente para codificar las variaciones de presión del aire captadas por un dispositivo de adquisición. El audio representado de esta manera se denomina **"forma de onda"** (figuras 2.1 y 2.2, superior), ya que es una discretización de la onda sonora, con cierta frecuencia de muestreo y número de bits por muestra.

Las **frecuencias de muestreo** utilizadas en audio digital dependen de la aplicación, pero el rango típico abarca desde los 8000 Hz (telefonía) hasta los 44100 Hz (calidad CD), esto es, **miles de muestras por segundo**. Es por este motivo que la representación en forma de onda rara vez se utiliza como dato de entrada a sistemas de segmentación o clasificación. Uno de los pasos previos es, por norma general, la **extracción de características**, con la cual se consigue **reducir la dimensionalidad** de los datos **manteniendo la información** de interés.

Representación temporal y espectral

La forma de onda es una **representación temporal** de la señal acústica. Al enfrentarse a las tareas de segmentación o clasificación, conviene obtener una representación de la señal en el **dominio espectral**, es decir, la información de la energía y la fase con la que contribuye cada componente frecuencial a la señal. El paso al dominio espectral se realiza de manera tradicional con la **Transformada de Fourier** en su versión discreta (cuya implementación eficiente se conoce como FFT, *Fast Fourier Transform*), existiendo otras opciones como la CQT (*Constant Q Transform*) o las transformadas Wavelet (WT, *Wavelet Transform*).

$$\mathbf{FFT}\{x[n]\} \equiv X[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{2\pi i}{N} kn}, \qquad k = 0, \ldots, N-1 \tag{2.1}$$

Espectrograma

Ya que una de las características de las señales de voz y de música es que **sólo muestran estacionariedad a corto plazo**, son de especial interés las representaciones de la señal que muestran la **evolución temporal del espectro** de la señal. Esta es la finalidad de la **Transformada de Fourier de tiempo corto** (STFT, *Short-Time Fourier Transform*).

Esta transformada divide la señal temporal en pequeños tramos (de duración y solapamiento determinados), y las multiplica por una función $w[n]$ (proceso de enventanado), y aplica sobre cada tramo resultante la FFT. El propósito de la **función de enventanado**, $w[n]$, es suavizar las transiciones abruptas del inicio y el final de la ventana, evitando así la aparición de frecuencias no deseadas.

$$\mathbf{STFT}\{x[n]\} \equiv X[m,k] = \mathbf{FFT}\{x[n]w[n-m]\} = \sum_{n=0}^{N-1} x[n]w[n-m] e^{-\frac{2\pi i}{N} kn} \tag{2.2}$$

El resultado es una **señal bidimensional**, interpretable como una matriz en la que cada columna contiene los valores de la FFT de una ventana de la señal original. El módulo de esta matriz (o su cuadrado) se conoce como el **espectrograma** de la señal (figuras 2.1 y 2.2, central), y es la representación más habitual de la **evolución temporal de las componentes espectrales**.

La escala mel

Las transformadas tradicionales asignan la misma importancia a todo el rango de frecuencias. Sin embargo, esta no es la forma en la que funciona la percepción

auditiva humana. Este hecho ha motivado la investigación de **modelos espectrales perceptuales**, es decir, basados en el funcionamiento del sistema auditivo humano.

Uno de estos modelos es la **escala mel**, propuesta en el año 1937 por Stevens, Volkman y Newmann [20]. Esta escala se desarrolló mediante experimentos empíricos en los que distintos oyentes señalaban qué frecuencias percibían como **equiespaciadas**. La fórmula utilizada para la conversión de hercios a mels es:

$$m_{mels} = 1127,01048 \ln(1 + f_{Hz}/700) \tag{2.3}$$

En la práctica, el uso de la escala mel implica una **compresión de la parte alta del espectro** y una **expansión de la resolución en bajas frecuencias**.

La escala mel puede ser aplicada a una representación tiempo-frecuencia, similar al espectrograma. En este caso, sobre cada ventana de la señal temporal se aplica un **banco de filtros adaptado a la escala mel** (conocidos como ***mel-filters***). En la matriz resultante (denominada **melgrama**) (figuras 2.1 y 2.2, inferior), cada columna contiene la energías de las señales resultantes tras el paso de la ventana temporal por cada uno de los filtros.

Alternativamente, se puede aplicar la transformación correspondiente a cada *mel-filter* sobre las frecuencias de un espectrograma obtenido previamente.

2.2. Modelos basados en aprendizaje automático

El auge de las técnicas de aprendizaje automático ha supuesto un **cambio de paradigma** en muchas áreas de investigación relacionadas con la inteligencia computacional, como pueden ser el procesamiento de lenguaje natural, la visión por computador o las tareas de segmentación y clasificación de audio ya citadas.

Una de las **diferencias clave** respecto al enfoque clásico de dichas tareas es el **aprendizaje a partir de los datos**: dado un conjunto de datos y, en el caso del aprendizaje supervisado, sus correspondientes **etiquetas** (la información de interés: qué clase de contenido se puede encontrar en el audio, nombre del locutor, idioma...), y definiendo una **función de coste**, la búsqueda del mejor modelo se convierte en un **problema de optimización** (generalmente, minimización) de esta función.

Aunque en la elaboración de los modelos considerados “clásicos” también hay parámetros ajustables mediante la minimización de una función de coste, no es una parte esencial de la construcción del modelo, como sí lo es, por ejemplo, en las redes neuronales.

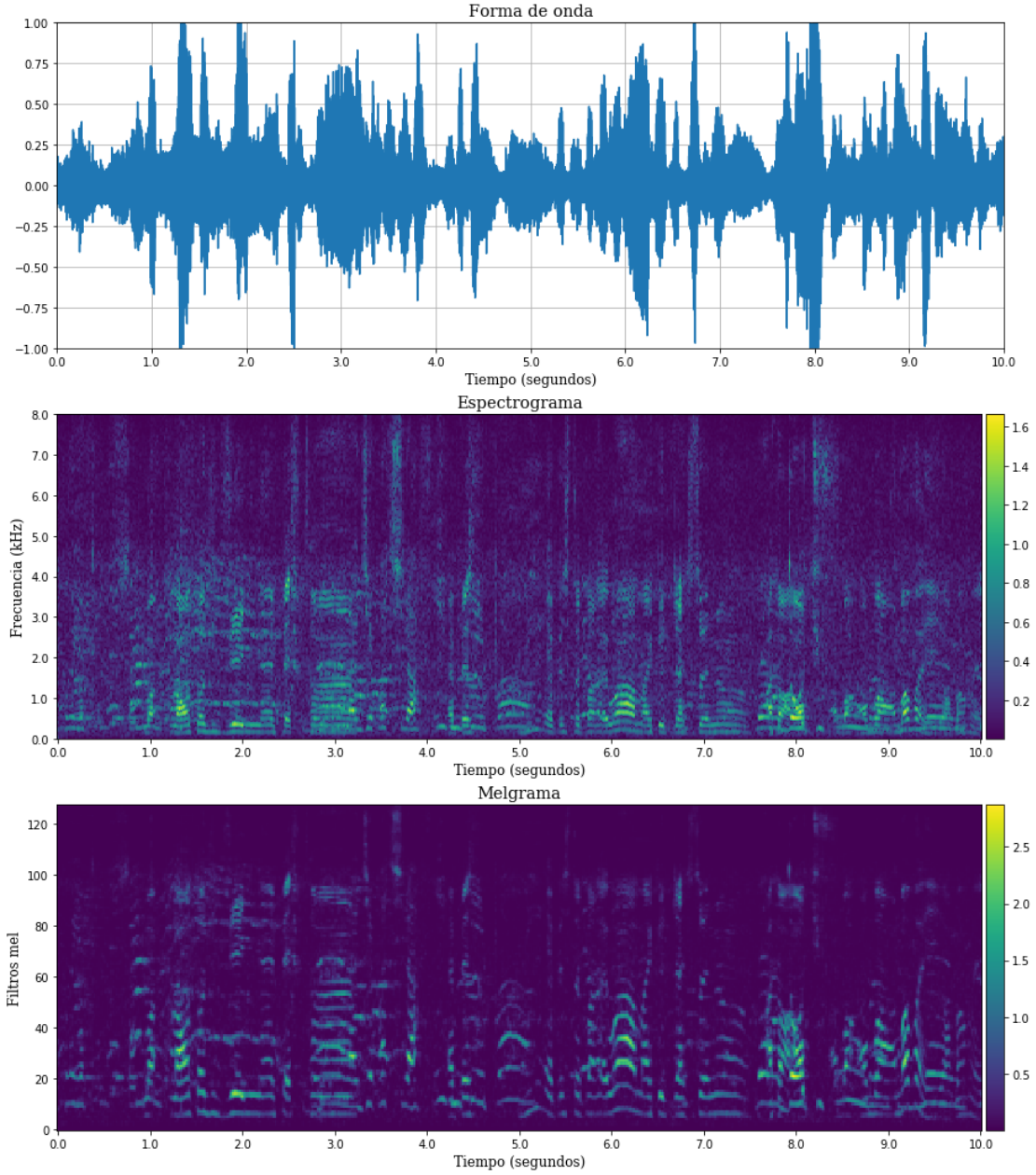

Figura 2.1: Forma de onda, espectrograma y melgrama correspondientes a una señal de audio que contiene voz (extraída de AudioSet)

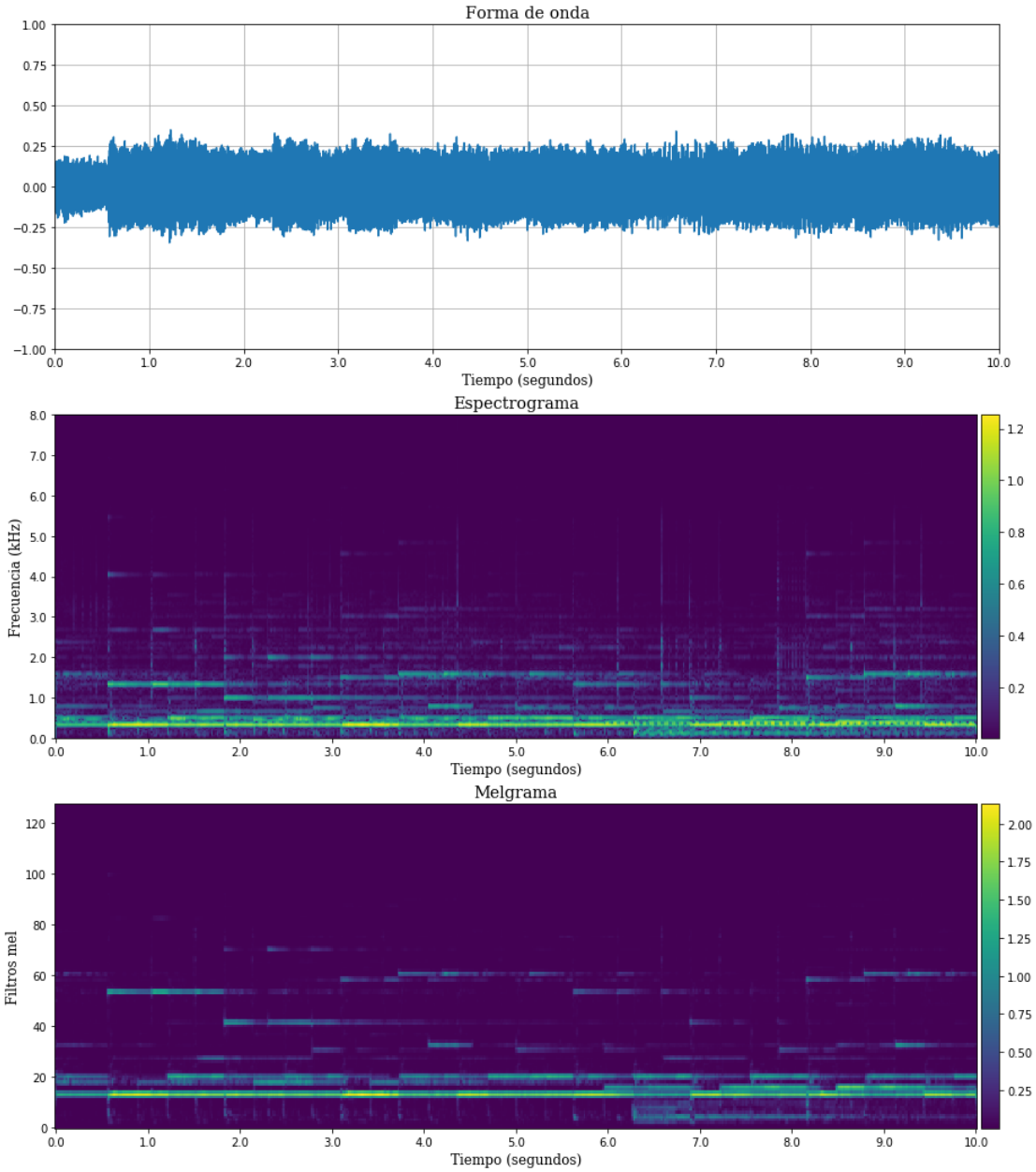

Figura 2.2: Forma de onda, espectrograma y melgrama correspondientes a una señal de audio que contiene música (extraída de AudioSet)

2.2.1. Redes neuronales

Una **red neuronal** (o red neuronal artificial) es un modelo de aprendizaje automático basado en la conexión de distintas **unidades de cálculo** (denominadas **neuronas**) distribuidas en **varias capas**. Cada neurona consta de valores de entrada, que llegan de la capa anterior, y valores de salida, que se propagan hacia la capa siguiente.

La primera capa de la red es la capa de **entrada**, y su dimensionalidad ha de ajustarse a la de los datos a procesar. La última capa es la de **salida**, en la cual el número de neuronas dependerá del problema a resolver (por ejemplo, del número de categorías en la clasificación).

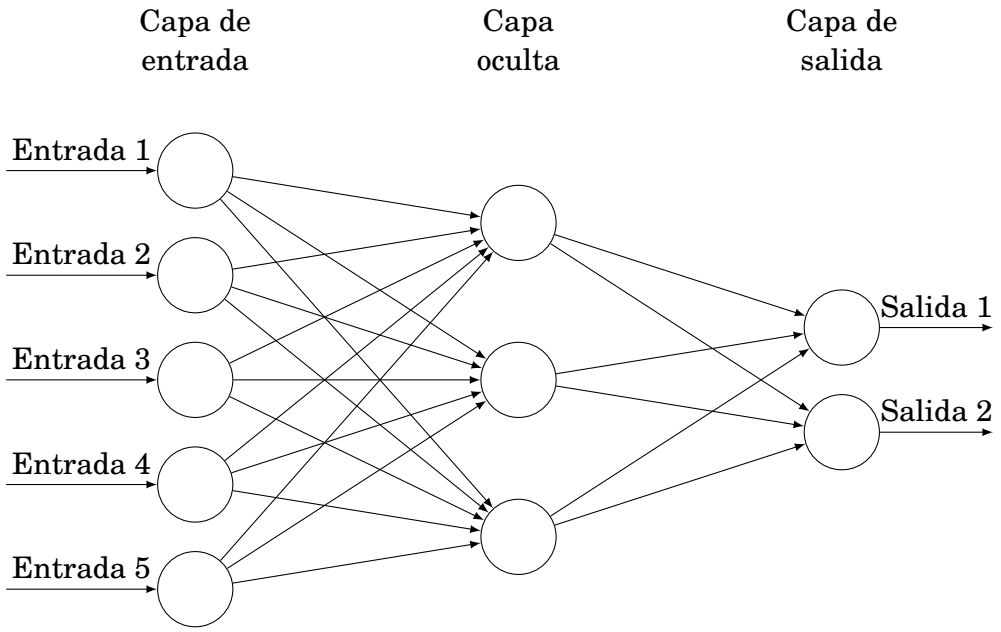

Figura 2.3: Estructura de una red neuronal *fully-connected* con una capa oculta

El número de **capas ocultas**, situadas entre las de entrada y salida, y el **número de neuronas** de cada una de ellas son **hiperparámetros** del diseño del modelo, al igual que las funciones de activación de las neuronas. Esto, sumado a los **diferentes tipos de capa** existentes (*fully-connected*, convolucionales, LSTM, entre otras), la tasa de aprendizaje o el uso de técnicas de regularización, supone una **multitud de decisiones de diseño** que complican la búsqueda del mejor modelo.

Capas *fully-connected* (DNN)

Las capas totalmente conectadas, o ***fully-connected***, dan lugar a la **estructura más básica** de redes neuronales (figura 2.3). Las salidas se calculan a partir de los valores de entrada (x_i), ponderados por **pesos entrenables** (w_i) y sumados a un valor de ***bias*** (b), aplicados posteriormente como argumento a una **función de activación**, f (figura 2.4).

$$n = \sum_i (w_i x_i) + b \tag{2.4}$$

$$y = f(n) = f(\sum_i (w_i x_i) + b) \tag{2.5}$$

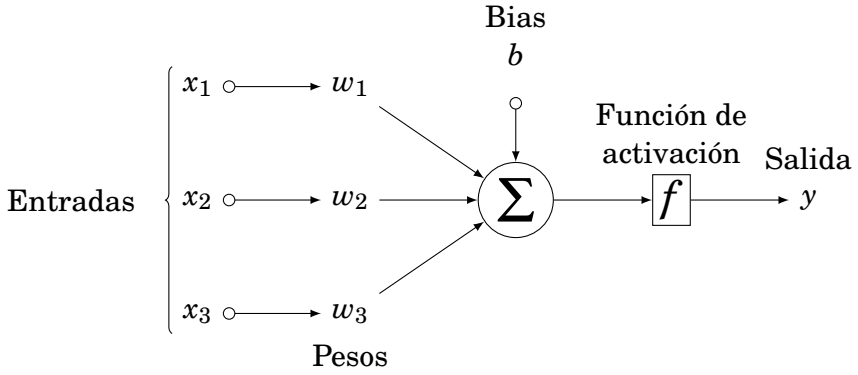

Figura 2.4: Diagrama del cálculo de la salida de una neurona en una capa DNN

Esta estructura es equivalente al modelo de aprendizaje del perceptrón multicapa (MLP, *Multi-Layer Perceptron*), y se conoce como red *fully-connected* o **DNN** (*Dense Neural Network*). Además, estas capas pueden utilizarse como parte de arquitecturas más complejas.

Capas convolucionales (CNN)

En muchas ocasiones, los datos de entrada son secuencias que muestran cierta **estructura espacial o temporal**. Esto ocurre en las señales de audio, que generalmente entran a la red como una matriz bidimensional, con un eje temporal y otro frecuencial. Es análogo el caso de las **redes aplicadas a visión por computador**, en las que **las entradas son imágenes** dispuestas en una matriz bidimensional (en tres canales, en el caso de imágenes a color).

Para aprovechar este tipo de estructuras en los datos surgen las capas convolucionales, que dan lugar a estructuras de **redes convolucionales** (CNN, *Convolutional Neural Networks*). Cada neurona de una capa convolucional es un **filtro**

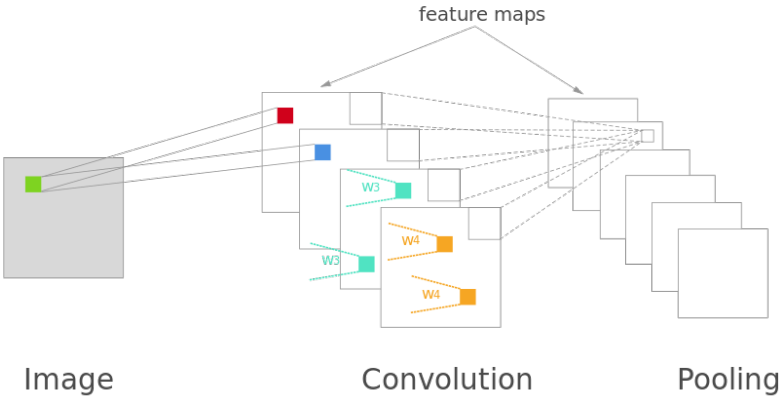

Figura 2.5: Transformaciones aplicadas a una matriz a lo largo de una red convolucional típica

(o *kernel*) de dimensión prefijada, cuyos pesos son entrenables. La salida es el resultado de **aplicar una convolución con este filtro a la matriz de entrada**.

El aprovechamiento de la información de los valores adyacentes en los datos de entrada es posible gracias a la operación de convolución. En una capa de K filtros cuadrados de dimensión $(2d+1)\times(2d+1)$, esta operación se llevaría a cabo como:

$$Y_k[i,j] = \sum_{n=i-d}^{i+d} \sum_{m=j-d}^{j+d} X[n,m]\, H_k[n-i, m-j], \quad k = 1,\ldots,K \tag{2.6}$$

Así, una capa convolucional tiene **una salida matricial por cada uno de sus filtros**. Esto permite la aplicación consecutiva de varias capas convolucionales. Adicionalmente, estas capas suelen intercalarse con las llamadas **capas de *pooling***, cuyo propósito es **submuestrar los datos** mediante cierto criterio (por ejemplo, **MaxPooling**: dividir en bloques y escoger los valores máximos).

Una estructura CNN puede contar, además, con una capa *fully-connected* previa a la capa de salida.

Capas LSTM

El procesamiento de **secuencias temporales** mediante redes neuronales puramente *fully-connected* no aprovecha la información de la estructura temporal de los datos. Para solventar este problema, se idearon las **capas RNN**, *Recurrent Neural Network* [21], en las que **cada neurona recibe información** del dato correspondiente al instante presente, pero también **de los datos previos en el tiempo**. Así se consigue dotar de cierta **memoria** al modelo.

Las capas RNN, sin embargo, muestran como punto débil el **corto plazo** de

su memoria, debido al **desvanecimiento del gradiente** [22, 23]. Para mejorar este aspecto, se proponen las **capas LSTM** (*Long Short-Term Memory*) [24], cuya memoria es potencialmente infinita gracias a que pueden **aprender qué datos olvidar**, qué datos **recordar**, y cuándo emplearlos. Esto se consigue mediante pesos entrenables en la **puerta de entrada**, la **puerta de salida** y la **puerta de olvido**.

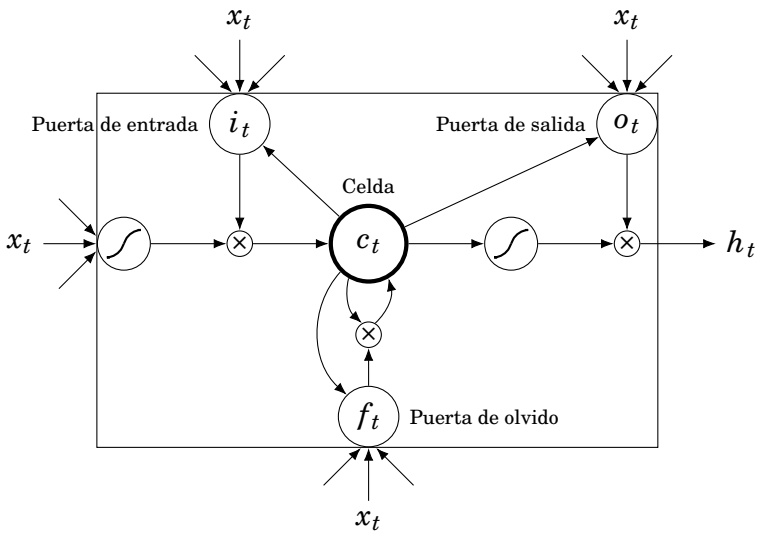

Figura 2.6: Diagrama de una neurona en una capa LSTM

La salida de una capa LSTM es otra secuencia temporal, de forma que es posible diseñar modelos con **varias capas LSTM consecutivas**, o tomar como resultado el último valor de cada secuencia.

Funciones de activación

La **función de activación** de una capa define el rango de posibles salidas de sus neuronas, y su primera derivada tiene un papel clave en la optimización de la función de coste. Las funciones de activación más comunes en redes neuronales son la **función sigmoide**, la **tangente hiperbólica**, **SoftMax** y **ReLU**.

La **función sigmoide**, o función logística, es de uso común en otros modelos de aprendizaje automático como la **regresión logística**. Su salida está acotada entre 0 y 1, es monótona creciente y su derivada toma valores mayores para los argumentos más cercanos a 0.

$$P(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

La **tangente hiperbólica** tiene propiedades similares a la sigmoide, pero su rango de salida es $(-1,1)$.

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.8}$$

La **función SoftMax**, también conocida como exponencial normalizada, es una generalización de la función logística a **vectores multidimensionales**. Para un vector de dimensión K:

$$\sigma(\mathbf{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \qquad j = 1,\ldots,K \tag{2.9}$$

Las salidas de las neuronas de una capa que usa SoftMax como función de activación están acotadas entre 0 y 1, y además **todas las salidas de la capa suman 1**. Esta propiedad hace que sea una función adecuada para representar **funciones de densidad de probabilidad**, por eso su aplicación en redes neuronales suele ser la activación de la capa de salida de un problema de clasificación.

La función **ReLU** (*Rectified Linear Unit*) ha sido propuesta más recientemente con resultados muy llamativos. Esta función tiene por salida 0 para argumentos negativos, y la identidad para argumentos positivos.

$$\mathbf{ReLU}(x) = \max(0,x) \tag{2.10}$$

A diferencia de las funciones anteriores, **la derivada de la función ReLU no es continua**, ya que no es derivable en $x = 0$. Sobre el papel, esto es un impedimento para los algoritmos de optimización, pero en la práctica no existe tal problema. Al contrario, la sencillez del cálculo de su derivada conlleva **mejoras de rendimiento** que han popularizado su uso.

$$\frac{d}{dx}\mathbf{ReLU}(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases} \tag{2.11}$$

Otra diferencia de la función ReLU respecto al resto de funciones descritas es que su salida no tiene cota superior (si la entrada no está acotada).

2.2.2. Funciones de coste y optimización

En aprendizaje supervisado, **una función de coste cuantifica la idoneidad de un modelo** en función de sus **predicciones** para los datos ($\hat{\mathbf{y}}$) y las **respuestas reales** ($\mathbf{y}$). El valor tomado por una función de coste será **menor cuanto mejor se ajusten** las predicciones del modelo a las etiquetas conocidas.

Para poder llevar a cabo el proceso de optimización, la función de coste debe ser **derivable respecto a los parámetros del modelo** (los pesos de las neuronas).

En problemas de **clasificación**, la función de coste más usual es la **entropía cruzada** entre las respuestas reales, y, y las predicciones expresadas como una distribución de probabilidad, $\hat{y}$.

Un modelo con **menor entropía cruzada** no necesariamente tiene una tasa de acierto mayor para un mismo conjunto de datos, pero da **distribuciones de probabilidad más aproximadas** a las etiquetas. Por ejemplo, un dato para el que se predice una probabilidad del 90 % de pertenecer a su clase correcta aporta menos a la entropía cruzada que un dato para el cual se predice un 60 %, sin embargo ambos cuentan de manera idéntica para la tasa de acierto.

$$H(y,\hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i \tag{2.12}$$

$$H(\mathbf{y},\hat{\mathbf{y}}) = \sum_i H(y_i,\hat{y}_i) \tag{2.13}$$

En el caso de problemas de **regresión**, las salidas esperadas no son categóricas, sino que tienen un rango continuo. Por ello, se utilizan funciones de coste diferentes, como el **MSE** (*Mean Square Error*) o el **MAE** (*Mean Absolute Error*).

$$\mathbf{MSE}(\mathbf{y},\hat{\mathbf{y}}) = \frac{1}{N}\sum_{i=1}^{N}(y-\hat{y})^2 \tag{2.14}$$

$$\mathbf{MAE}(\mathbf{y},\hat{\mathbf{y}}) = \frac{1}{N}\sum_{i=1}^{N}|y-\hat{y}| \tag{2.15}$$

La técnica básica para el aprendizaje en redes neuronales es el algoritmo de retropropagación, o ***back-propagation*** [25]. Este algoritmo **estima el gradiente del error a la salida** (dado por la función de coste) **respecto a los pesos**, que entonces pueden ser actualizados en cada iteración por algoritmos de optimización como el descenso por gradiente.

La idea tras el **descenso por gradiente** es que, pudiendo hallar tras una iteración las derivadas parciales de la función de coste respecto a cada uno de los parámetros, de cara a la siguiente iteración podremos **actualizar estos parámetros** en la dirección que **minimice el coste**.

Siendo W^t el vector de pesos o parámetros del modelo en la iteración t, y $L(W)$ la función de coste escogida, los pesos para la siguiente iteración son escogidos como:

$$W^{t+1} = W^t - \rho \nabla L(W) \tag{2.16}$$

El parámetro ρ es la **tasa de aprendizaje** (*learning rate*), y regula la velocidad de actualización de los pesos en cada iteración. La tasa de aprendizaje es un parámetro crítico para que el entrenamiento del modelo converja.

La optimización basada en **descenso por gradiente** ha recibido diversas modificaciones y revisiones motivadas por el interés en las redes neuronales. Una de las más recientes y populares es el **algoritmo Adam** [26].

El objetivo de Adam es **mejorar la eficiencia de la optimización** para adecuarse a las necesidades de los modelos actuales, cada vez más grandes y con más datos. Para ello, estima de forma individual para cada parámetro una **tasa de aprendizaje adaptativa**, a partir de los momentos de primer y segundo orden del gradiente de la función de coste.

2.3. Reconocimiento de eventos en señales de audio

Durante el presente capítulo se han citado líneas de investigación en el procesamiento de audio como la detección de idioma, la segmentación por locutor o la detección de actividad de voz y de música. Los experimentos llevados a cabo en este trabajo se enmarcan bajo la tarea de **reconocimiento de eventos acústicos en señales de audio**, relacionada con las anteriores pero con un planteamiento diferente.

Se define como reconocimiento de eventos acústicos la capacidad de **identificar sucesos a partir del sonido que producen**. Por ejemplo, al romperse un cristal, un oyente puede hacerse una idea de lo ocurrido a partir del sonido que ha escuchado, **sin necesidad de información visual** o de otros tipos. Una gran cantidad de eventos producen **sonidos característicos** que nos resultan suficientes para identificarlos: la lluvia, el ladrido de un perro, la voz humana o un instrumento musical.

Aplicar este concepto al campo de la inteligencia computacional es un problema comparable al de reconocimiento de objetos en **visión por computador**. No sólo son **tareas similares conceptualmente**, sino que, al disponer las señales de audio en matrices de dos dimensiones (tiempo y frecuencia) mediante representaciones como el espectrograma o el melgrama, también el **formato de los datos** es semejante. De esta forma, tiene sentido pensar que los **avances en visión por computador** conseguidos en los últimos años [27,28] pueden guiar el progreso de las investigaciones en **reconocimiento de eventos acústicos**.

La investigación en este campo ha dado lugar a evaluaciones como la del proyecto CHIL (*Computers in Human Interaction Loop*) [29] o DCASE (*Detection and Classification of Acoustic Scenes and Events*) [10]. Con el fin de **impulsar el desarrollo de modelos** para la detección de eventos de audio a gran escala, Google desarrolla la ontología y base de datos **AudioSet** [1] a partir de **segmentos de audio extraídos de vídeos** de la plataforma YouTube.

Google AudioSet propone una ontología para la clasificación de eventos de audio con un total de **632 categorías**, organizadas en una jerarquía de clases y subclases. Las características de esta ontología, su definición y su uso en el trabajo son detallados en la sección 3.1.

Capítulo 3

Descripción y diseño de los experimentos

3.1. Entorno experimental: Google AudioSet

Los **ficheros de audio** y las **etiquetas** empleadas durante los experimentos son un subconjunto de los que se pueden encontrar en **Google AudioSet**[1].

3.1.1. Especificación de la ontología

AudioSet consta, por un lado, de una **ontología** de **632 categorías** particularmente **enfocadas al reconocimiento de eventos de audio**. Se evita especificar en las categorías toda aquella información que no se pueda adquirir a partir del sonido. Por ejemplo, un evento de la categoría "golpe" es clasificado como tal, aunque una descripción más precisa fuera "pie descalzo pisando un suelo de madera".

56 de las categorías pertenecen a una "lista negra" de etiquetas que se han considerado poco convenientes para las tareas de clasificación (por ejemplo, una de las categorías de más alto nivel, "sonidos de cosas"), y 22 son consideradas "abstractas". Estas últimas son categorías que ayudan a estructurar la ontología pero no están pensadas para ser utilizadas como etiquetas (por ejemplo, "onomatopeya").

La **especificación de la ontología** está contenida en un fichero JSON, con una entrada por categoría en la que se detallan:

- **`id`:** Cadena de texto que sirve como identificador único para la categoría. El identificador asignado es el *Knowledge Graph Machine ID* [30] (MID) de Google más adecuado para la categoría.
- **`name`:** El nombre descriptivo de la categoría, en inglés.

[1] http://g.co/audioset

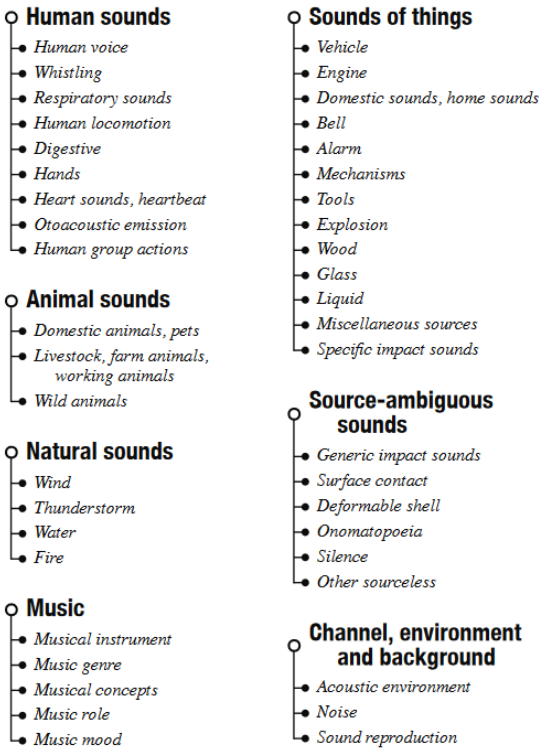

Figura 3.1: Categorías pertenecientes a la ontología AudioSet, extraída de [1]

- **`description`:** Una o dos frases que explican el significado de la categoría.
- **`citation_uri`:** Un enlace externo a Wikipedia o WordNet pesnado para enfatizar el uso específico de la categoría sobre eventos de audio.
- **`positive_examples`:** Lista de enlaces a segmentos de vídeos de YouTube cuyo audio contiene algún evento perteneciente a esta categoría.
- **`childs_ids`:** Lista de identificadores de las subcategorías.
- **`restrictions`:** Observaciones como la pertenencia a la lista negra (`blacklisted`) o la abstracción (`abstract`).

3.1.2. Especificación del conjunto de datos

El **conjunto de datos original** proporcionado por AudioSet y descrito en [1] consiste en **1.789.621 segmentos de audio** con una duración de **diez segundos** cada uno, sumando un total de **4.971 horas**. Los segmentos están extraídos de

```
"id": "/m/09x0r",
"name": "Speech",
"description": "Speech is the vocalized form of human communication, created out of the phonetic
     combination of a limited set of vowel and consonant speech sound units.",
"citation_uri": "http://en.wikipedia.org/wiki/Speech",
"positive_examples": ["youtu.be/8uI9H5jGRV8?start=30&end=40", "youtu.be/cz8QIJHjZh4?start=30&end
    =40", "youtu.be/91FIbeKyIhw?start=30&end=40", "youtu.be/V3HdocEv8gw?start=220&end=230", "
    youtu.be/a8igK2oWlVY?start=30&end=40", "youtu.be/2hvTtM7VdJg?start=30&end=40", "youtu.be/
    S7ytQItOrkk?start=30&end=40", "youtu.be/Zyl2Q3GepPA?start=30&end=40", "youtu.be/CU1qRm_Bcps?
    start=30&end=40"],
"child_ids": ["/m/05zppz", "/m/02zsn", "/m/0ytgt", "/m/01h8n0", "/m/02qldy", "/m/0261r1", "/m/0
    brhx"],
"restrictions": []
```


Figura 3.2: Entrada de la categoría “Voz” (“Speech”) en la ontología AudioSet

vídeos de la plataforma YouTube y etiquetados por un jurado con las categorías descritas en la ontología que se adecúen a los contenidos del segmento de audio.

Los segmentos contenidos en AudioSet están organizados en **tres subconjuntos disjuntos**:

- **Entrenamiento equilibrado** (*Balanced train*): Conjunto de entrenamiento con segmentos seleccionados para conseguir una **representación equilibrada** de las clases.
- **Evaluación** (*Evaluation*): Conjunto de evaluación diseñado con el mismo criterio de **equilibrio de clases**.
- **Entrenamiento desequilibrado** (*Unbalanced train*): Conjunto de mucho mayor tamaño que contiene el resto de los segmentos no incluidos en los otros dos. Sin embargo, no tiene **ningún criterio de equilibrio** entre las clases.

En la **especificación original**, sólo dos de los conjuntos están definidos: el conjunto de evaluación (17.748 segmentos en dicha especificación) y el de entrenamiento desequilibrado (1.771.873 segmentos). El **estado actual** (**Junio ’18**) mostrado en la web de AudioSet describe un total de **2.085.544 segmentos** de audio: **20.383** en el conjunto de evaluación, **22.176** en el conjunto de entrenamiento equilibrado y **2.042.985** en el conjunto de entrenamiento desequilibrado. Se trata, por tanto, de un conjunto de datos todavía **en proceso de crecimiento** a pesar de su dimensión.

3.1.3. Obtención de los segmentos de audio

En AudioSet no se incluyen de forma directa los ficheros de audio, sólo se especifica para cada segmento el **enlace al vídeo** de YouTube que lo contiene y

	Original	Marzo '17	Junio '18	AUDIAS2018
Evaluación	17.748	20.371	20.383	19.542
Entrenamiento equil.	0	22.160	22.176	21.364
Entrenamiento deseq.	1.771.873	2.041.789	2.042.985	>37.030
Total	**1.789.621**	**2.084.320**	**2.085.544**	**>77.936**

Tabla 3.1: Número de segmentos contenidos en las diferentes especificaciones de AudioSet

la **ubicación temporal** dentro del vídeo. A partir de esta información, los ficheros de audio correspondientes a los segmentos de AudioSet han sido descargados mediante un script programado por el profesor Doroteo Torre.

Los **archivos de audio** correspondientes a los segmentos han sido obtenidos en **formato WAV-PCM** con **16 bits por muestra**, en estéreo y a una frecuencia de muestreo de **16.000 Hz**. En este formato, 10 segundos de audio quedan recogidos en **320.000 muestras**, codificadas en un total de **625 Kilobytes**.

El listado de AudioSet utilizado para **acceder a los ficheros** (**Marzo '17**) marca la existencia de **22.160** segmentos en el conjunto de entrenamiento equilibrado, **20.371** en el conjunto de evaluación y **2.041.789** en el conjunto de entrenamiento desequilibrado: una cantidad de segmentos **mayor que en la especificación original**, pero **menor que en la actual** (Junio '18).

Otro factor importante en la obtención de los ficheros es la **disponibilidad** de los mismos. Al depender de la plataforma YouTube para la descarga, **no está asegurado el acceso** a los ficheros: por ejemplo, el usuario que subió el vídeo podría haberlo retirado de la web. Esto explica que, tras la descarga, el total de ficheros descargados sea inferior al número de segmentos incluidos en la especificación.

En la tabla 3.1 se muestra el **número de segmentos de cada especificación** a la que se ha hecho referencia, siendo **AUDIAS2018** el conjunto de segmentos cuyos ficheros de audio han sido descargados.

3.1.4. Definición de las etiquetas utilizadas

Para entrenar y evaluar modelos de detección de voz y música sobre los segmentos obtenidos, primero es preciso obtener la información de **presencia de música** y **presencia de voz** a partir de las etiquetas de AudioSet. AudioSet tiene categorías correspondientes a **voz humana** (id = `/m/09x0r`) y **música** (id = `/m/04rlf`). Se han obtenido etiquetas binarias para voz y para música **comprobando la presencia de estas categorías** (o sus subcategorías) **en el etiquetado de cada segmento**.

	Sin música	Con música	Total
Sin voz	52,28%	21,57%	**73,85%**
Con voz	19,70%	6,45%	**26,15%**
Total	**71,98%**	**28,02%**	**100%**

Tabla 3.2: Distribución de voz y música en el subconjunto de entrenamiento equilibrado (AUDIAS2018)

	Sin música	Con música	Total
Sin voz	53,26%	20,35%	**73,61%**
Con voz	19,15%	7,24%	**26,39%**
Total	**72,41%**	**27,59%**	**100%**

Tabla 3.3: Distribución de voz y música en el subconjunto de evaluación (AUDIAS2018)

El etiquetado proporcionado por AudioSet indica **todas aquellas categorías que aparecen** en cada segmento al completo. No hay información acerca de la **ubicación temporal** de cada evento dentro del segmento, o sobre qué categorías cobran más importancia que otras: esto se conoce como **etiquetado débil**.

En las tablas 3.2, 3.3 y 3.4 se muestra la **distribución de las etiquetas de música y de voz** en los subconjuntos de **evaluación** y **entrenamiento equilibrado** de la especificación **AUDIAS2018** y en el subconjunto de **entrenamiento desequilibrado** de la especificación **Marzo '17**.

Tanto el subconjunto de evaluación como el de entrenamiento equilibrado, diseñados con el propósito de representar la totalidad de categorías presentes en AudioSet, quedan **fuertemente desequilibrados** al estudiar la distribución de las etiquetas de música y de voz: tanto música como voz tienen prioris inferiores al 30% en los dos conjuntos, y más de la mitad de los segmentos no pertenecen a ninguna de las dos categorías.

Por el contrario, el subconjunto desequilibrado muestra un **mayor equilibrio** entre las dos clases de interés: el 49,20% de los segmentos contienen voz, y el 48,95% contienen música.

3.1.5. Definición del conjunto de datos

El **desequilibrio de clases** es un **fenómeno a evitar** en el desarrollo de modelos de clasificación. La opción escogida para sortear este problema ha sido la

	Sin música	**Con música**	**Total**
Sin voz	17,30%	33,50%	**50,80%**
Con voz	33,76%	15,44%	**49,20%**
Total	**51,05%**	**48,95%**	**100%**

Tabla 3.4: Distribución de voz y música en el subconjunto de entrenamiento desequilibrado (Marzo '17)

	Sin música	**Con música**	**Total**
Sin voz	27,69%	24,10%	**51,79%**
Con voz	24,10%	24,10%	**48,21%**
Total	**51,79%**	**48,21%**	**100%**

Tabla 3.5: Distribución de voz y música en el conjunto AUDIAS-Junio'18

construcción de un conjunto de datos, **AUDIAS-Junio'18**, que incluye:

- El subconjunto de **entrenamiento equilibrado** de AUDIAS2018 (**21.364** segmentos).
- El subconjunto de **evaluación** de AUDIAS2018 (**19.542** segmentos).
- **37.030** segmentos del subconjunto de **entrenamiento desequilibrado**, escogidos como la mínima cantidad necesaria a añadir para **equilibrar las clases**.

El conjunto resultante logra **equilibrar las clases de interés** (tabla 3.5), y consiste en **77.936** segmentos de 10 segundos, es decir, **216 horas** (9 días) de audio. Este conjunto es de tamaño más que suficiente para el entrenamiento de modelos basados en redes neuronales, a pesar de que **sólo contiene un 3,73% del total** de segmentos de AudioSet (Marzo '17).

3.2. Extracción de características de audio

La entrada a los modelos de clasificación estudiados en este trabajo no ha sido la forma de onda de los segmentos de audio, sino una **representación bidimensional** (en tiempo y frecuencia) de los mismos basada en los filtros mel. Esta representación, conocida como **melgrama** o *mel-spectrogram*, ha sido extraída mediante la implementación en Python incluida en el módulo `librosa` [31].

El proceso de extracción del melgrama consiste en:

1. **Obtención del espectrograma** de la señal (monocanal) mediante la Short-Time Fourier Transform (STFT). En este paso se ha aplicado la FFT con 512 *bins* a ventanas de 512 muestras de audio (32 ms) con separación de 320 muestras (20 ms) entre ventanas consecutivas.

2. **Transformación** de las frecuencias del espectrograma al **melgrama** a partir de un banco de **128 filtros mel**.

La representación resultante para cada uno de los segmentos es una **matriz de tamaño 128 × 500**, donde la primera dimensión es la **frecuencial** (128 filtros mel) y la segunda es la **evolución temporal** (500 pasos de 20 milisegundos). Cada matriz contiene, entonces, 64.000 muestras, un 20 % de las que contenía la forma de onda original.

3.3. Modelos de clasificación propuestos

Los **modelos de clasificación** evaluados durante este trabajo consisten en diferentes **estructuras de redes neuronales** entrenadas para diferenciar:

a) Segmentos que contienen **voz** y segmentos que no la contienen (**dos clases**).

b) Segmentos que contienen **música** y segmentos que no la contienen (**dos clases**).

c) Las **cuatro clases** obtenidas como **combinación** de las descritas en a) y b):

   1. Segmentos que **no contienen música ni voz**.
   2. Segmentos que **contienen voz** pero **no música**.
   3. Segmentos que **contienen música** pero **no voz**.
   4. Segmentos que **contienen tanto música como voz**.

   Se hará referencia a esta tarea como “detección conjunta de voz y de música”.

Las estructuras propuestas son diferentes configuraciones de **redes *fully-connected*** (DNN), **redes convolucionales** (CNN) y **redes LSTM**, así como redes neuronales que combinan varias de ellas. Todas ellas reciben como entrada la matriz bidimensional definida en la sección 3.2.

3.3.1. Modelos basados en arquitecturas DNN

Los modelos DNN se han diseñado parametrizando tanto el **número de capas ocultas** (L) como el **número de neuronas** de cada capa oculta (N). De esta forma, estos modelos cuentan con:

- Una **capa de entrada**, que recibe las matrices de datos (los **melgramas**, de tamaño 128×500 por cada segmento).
- L **capas ocultas *Dense*** de N neuronas cada una, con función de activación **ReLU**.
- Una **capa de salida** con activación **SoftMax** y tantas neuronas como clases (2 ó 4).

Al proporcionar una matriz bidimensional como entrada a una capa *Dense*, cada posición de la matriz se considera una característica aislada. Esto **impide**, en este tipo de modelos, el aprovechamiento de la información del **contexto temporal o frecuencial**.

El **número de capas ocultas**, L, ha tomado durante los experimentos los valores $[2,3,4,5,6]$. El rango de valores para el **número de neuronas** en cada capa, N, ha sido $[16,32,64,128,256,512,1024,2048]$.

Por ser el modelo más básico, **se ha empleado como sistema *baseline***, es decir, sus resultados han servido como contexto para interpretar el rendimiento del resto de modelos.

3.3.2. Modelos basados en arquitecturas CNN

Las **redes convolucionales** introducen parámetros de diseño adicionales frente a las DNN, como la **dimensión de los filtros** o la **introducción de capas de *pooling*** para reducción de dimensionalidad. Para definir una búsqueda asequible en el espacio de hiperparámetros, se ha fijado la dimensión de los filtros a dos posibilidades, 3×3 (**CNN3x3**) y 7×7 (**CNN7x7**), y se ha introducido tras cada capa convolucional una **capa *MaxPooling2D*** con tamaño de rejilla 2×2, manteniendo los parámetros L (número de capas ocultas) y N (número de filtros en cada capa).

Adicionalmente, las arquitecturas CNN introducen, antes de la capa de salida, una **etapa adimensional**, cuya implementación más básica es una capa ***Flatten*** y otra ***Dense***. Así, los modelos constan de:

- Una **capa de entrada**, que recibe las matrices de datos (de tamaño 128×500 por cada segmento).
- ***L* capas ocultas convolucionales** de N filtros cuadrados (3×3 o 7×7) cada una, con función de activación **ReLU**.

- Una **capa de *max-pooling*** tras cada una de las L capas convolucionales, con tamaño de *pooling* 2×2.

- Una **capa *Flatten*** seguida de una **capa *Dense*** de 512 unidades.

- Una **capa de salida** con activación **SoftMax** y tantas neuronas como clases (2 ó 4).

Tras cada capa de *pooling*, el tamaño de los datos se reduce en un factor de 2 a lo largo de cada dimensión, manteniendo el valor máximo de activación en cada vecindario 2×2. De esta manera, cada capa convolucional procesa matrices de **tamaño cuatro veces menor** que la anterior, y los filtros, aun manteniendo su dimensión, operan sobre un **contexto temporal y frecuencial más amplio**.

En estos modelos, se han probado los valores $L = [4,5,6,7]$ para el número de capas y $N = [32,64,128,256]$ para el número de filtros de cada capa. En el caso de las redes con filtros de tamaño 7×7, sólo se han entrenado las redes con $L = 6$ y $L = 7$.

3.3.3. Modelos basados en arquitecturas LSTM

De nuevo, las estructuras basadas en capas ocultas LSTM cuentan con más parámetros de diseño que una red ***fully-connected***. Para la búsqueda, se han parametrizado de nuevo el **número de capas**, L, y el **número de neuronas** por capa, N, dejando el resto de parámetros en los valores por defecto proporcionados por Keras.

- Una **capa de entrada**, que recibe las matrices de datos (de tamaño 128×500 por cada segmento).

- ***L* capas ocultas LSTM** de N neuronas cada una, con función de activación **sigmoide**.

- Una **capa de salida** con activación **SoftMax** y tantas neuronas como clases (2 ó 4).

En los modelos implementados, la última capa LSTM tiene como salida el **último valor de la secuencia** obtenida en cada una de las N neuronas. La salida de cada neurona en el resto de capas LSTM es la **secuencia temporal completa**.

3.3.4. Modelos basados en arquitecturas híbridas

Red convolucional + LSTM

Las capas LSTM que se describen en el apartado 3.3.3 procesan 128 secuencias temporales (una por cada *mel-filter*) de 500 pasos cada una.

Situando una **etapa convolucional** antes de las capas LSTM, se consigue que a la etapa LSTM lleguen **secuencias ya transformadas por la red** y, mediante las capas de *Pooling*, también de **menor dimensión**.

Se han probado las siguientes **estructuras CNN+LSTM**:

- **CNN unidimensional + LSTM (C1-LSTM):** La convolución unidimensional puede aplicarse sobre la dimensión frecuencial o sobre la temporal. Aplicar la convolución sobre el **eje de frecuencias** mantiene la **información temporal intacta** hasta la etapa LSTM, mientras que se reduce la dimensionalidad en frecuencia (es decir, el número de secuencias temporales).

- **CNN bidimensional + LSTM (C2-LSTM):** En este caso, la etapa convolucional transforma la matriz de entrada y reduce su tamaño en **ambos ejes**. La etapa convolucional también opera en la **dimensión temporal**, lo cual puede facilitar el trabajo de la etapa LSTM.

Para parametrizar el diseño de estas arquitecturas, se han colocado ***L* capas convolucionales** seguidas de una etapa LSTM de ***L* capas** en C1-LSTM, o de **una capa** en C2-LSTM, con el objetivo de obtener modelos de una complejidad razonable. Cada capa de ambas etapas contiene ***N* filtros/neuronas**, siendo los filtros de **tamaño 3** en el caso de las convolucionales unidimensionales y de **tamaño 3×3** en las bidimensionales.

Estos modelos también incluyen una **capa *Dense* con 512 neuronas** previa a la capa de salida.

3.4. Entrenamiento y selección de modelos

Los modelos han sido implementados mediante la librería **Keras** [32] sobre **TensorFlow** [33] en Python 3, utilizando procesamiento por **GPU** (*NVIDIA GeForce GTX 1080*).

Para llevar a cabo el entrenamiento y la evaluación de los modelos, se ha dividido el **conjunto AUDIAS-Junio'18** en **subconjuntos de entrenamiento** (**43.643** segmentos), **validación** (**10.910** segmentos) y **test** (**23.383** segmentos) con un criterio de **mantener la proporción de clases** indicada en la tabla 3.5.

La función de coste a minimizar es la **entropía cruzada** (ecuación 2.14). El proceso de **entrenamiento** de un modelo calcula, tras cada iteración sobre el con-

junto de entrenamiento (*epoch*), la **función de coste** obtenida **sobre el conjunto de validación**.

Al terminar el entrenamiento, se escoge como **mejor modelo** aquel que haya conseguido el **mínimo coste en validación**, independientemente de su rendimiento en el conjunto de entrenamiento. Para este modelo, se calcula el **coste sobre el conjunto de test**, como corroboración del rendimiento obtenido en validación.

El procedimiento de **selección de modelos** permite señalar de manera empírica la **mejor configuración** de los hiperparámetros: desde la tasa de aprendizaje o el tamaño de *batch* hasta el número de capas y neuronas o, de manera más amplia, la estructura de la red. Como el tamaño del espacio de hiperparámetros hace inviable su exploración completa, se han definido valores discretos de los mismos para llevar a cabo una **búsqueda en rejilla** (*grid search*). La búsqueda en rejilla consiste en probar las configuraciones de hiperparámetros en cierto rango y escoger aquella que cumpla cierto criterio, en este caso, **minimizar la función de coste** en el conjunto de validación.

Un **proceso de entrenamiento típico** alcanza un punto en el cual el coste sobre el conjunto de entrenamiento sigue descendiendo, pero el coste en validación comienza a crecer. A partir de este punto, el entrenamiento sólo consigue que el modelo almacene **características concretas** de los datos de entrenamiento que **no son generalizables** a otros conjuntos (fenómeno de ***overfitting*** o **sobreentrenamiento**). Por eso, tras dicho punto el entrenamiento puede darse por finalizado: es el propósito del ***early stopping*** [34]. Durante los experimentos realizados se ha aplicado *early stopping* con el fin de acortar el tiempo de entrenamiento.

Para mayor interpretabilidad del rendimiento de cada modelo, se proporciona también la **precisión** o *accuracy* de cada uno de ellos (es decir, el porcentaje de acierto en clasificación). Esta medida, sin embargo, **no es utilizada para selección de modelos**.

Capítulo 4

Resultados

4.1. Descripción de los resultados obtenidos

Durante los experimentos llevados a cabo, se han entrenado por separado redes para la **detección de presencia de voz** y para la **detección de presencia de música**, además de redes que enfrentan **ambas tareas** simultáneamente (detección conjunta de voz y de música). En los apartados 4.1.1, 4.1.2 y 4.1.3 se presentan los resultados obtenidos respectivamente para cada uno de estos planteamientos.

Como primer paso, se han escogido de manera empírica valores para la **tasa de aprendizaje** y para el **tamaño de *batch***. Para ello, se han probado en cada uno de ellos los siguientes rangos de valores:

- **Tasa de aprendizaje:** $[10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$. El valor escogido es $\mathbf{10^{-4}}$.
- **Tamaño de *batch*:** $[16, 32, 64, 128, 256, 512]$. El valor escogido es **128**, si bien la complejidad de los modelos ha hecho necesaria su reducción en el caso de las redes convolucionales.

El criterio de selección de la **tasa de aprendizaje** tiene que ver con la correcta minimización de la función de coste, pero también con el tiempo de entrenamiento. Con tasas inferiores a 10^{-4}, se han obtenido rendimientos similares pero con un entrenamiento mucho más lento.

En cuanto al **tamaño de *batch***, el efecto en el rendimiento del modelo es menos crítico, pero tiene más influencia en el coste computacional del entrenamiento. Usar un tamaño mayor permite mejorar la eficiencia, pero también requiere un mayor uso de memoria que no es viable en todos los modelos [35, 36].

4.1.1. Resultados de redes neuronales para detección de voz

En la tabla 4.1 se muestran los mejores resultados obtenidos con cada modelo en la tarea de **detección de voz**, en términos de **coste** (entropía cruzada) y **tasa de acierto** (*accuracy*) para cada uno de los tres subconjuntos. Además, se especifican los **hiperparámetros** de diseño (número de capas, L, y número de neuronas por capa, N) con los que se ha logrado el mejor resultado en cada uno de los modelos. Se aporta también una comparación de la **complejidad** de cada modelo mediante la columna $p = \log_{10}(\text{nº de parámetros})$.

Modelo	L	N	p	Entrenamiento		Validación		Test	
				Coste	*Acc.*	Coste	*Acc.*	Coste	*Acc.*
DNN	6	512	6,2307	0,4887	77,03%	0,5100	76,45%	0,5180	75,58%
CNN3x3	7	128	6,0354	0,3222	86,86%	0,3832	83,65%	0,3868	83,72%
CNN7x7	6	64	6,1665	0,3615	85,02%	0,3803	84,07%	0,3899	83,21%
LSTM	1	64	4,6972	0,5472	73,69%	0,5438	73,51%	0,5472	73,41%
C1-LSTM	3	256	6,4034	0,4057	82,56%	0,4355	80,96%	0,4370	80,80%
C2-LSTM	**6**	**256**	**6,5882**	**0,3772**	**84,30%**	**0,3748**	**84,34%**	**0,3815**	**83,99%**

Tabla 4.1: Resultados en distintos modelos para detección de voz

El **mejor resultado en validación** es obtenido por la **red C2-LSTM** de **6 capas convolucionales**, con **256 neuronas** cada una de ellas. El tamaño de *batch* utilizado es 8, por limitaciones de memoria durante el entrenamiento. Esta red consigue un **84,34% de acierto en validación** y un **83,99% en test**. Nótese que este resultado es **muy similar** al conseguido con una red CNN con 6 capas de 64 filtros de 7×7, que logra una tasa de acierto del **84,07% en validación** y del **83,21% en test**.

La figura 4.1 es una **representación gráfica del rendimiento** de los modelos entrenados frente a su **número de parámetros**. Esta gráfica deja ver rangos de rendimiento muy diferentes entre algunos modelos, por ejemplo entre DNN y CNN, y muy leves entre las CNN con filtros de tamaño 3×3 y 7×7 y las C2-LSTM.

En la figura 4.2 se muestra la **entropía cruzada en validación** de las distintas arquitecturas entrenadas en función del número de capas y el número de neuronas de cada capa. Se marca con un punto blanco el **mejor resultado** hallado para cada arquitectura: estos son los resultados incluidos en la tabla 4.1.

El **proceso de entrenamiento** del mejor modelo hallado se ilustra en la figura 4.3, y la figura 4.4 muestra la **matriz de confusión** obtenida sobre el **conjunto de test**. En la matriz de confusión se indica el **porcentaje de segmentos de test** que corresponde a cada posible combinación *clase real-clase predicha*. De esta manera, en la matriz se pueden observar los **falsos positivos** (**10,41**% de los segmentos de test) y los **falsos negativos** (**5,60**% de los segmentos de test).

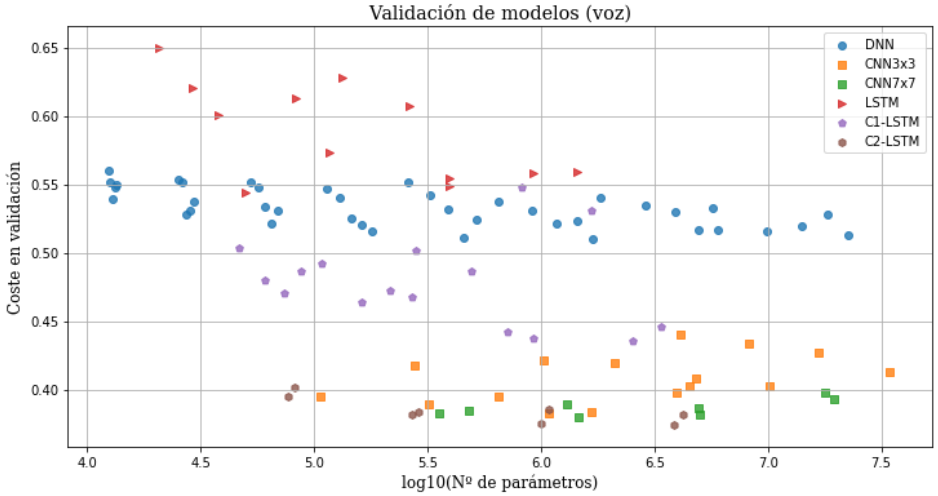

Figura 4.1: Resultados de validación de los modelos para detección de voz, frente al número de parámetros

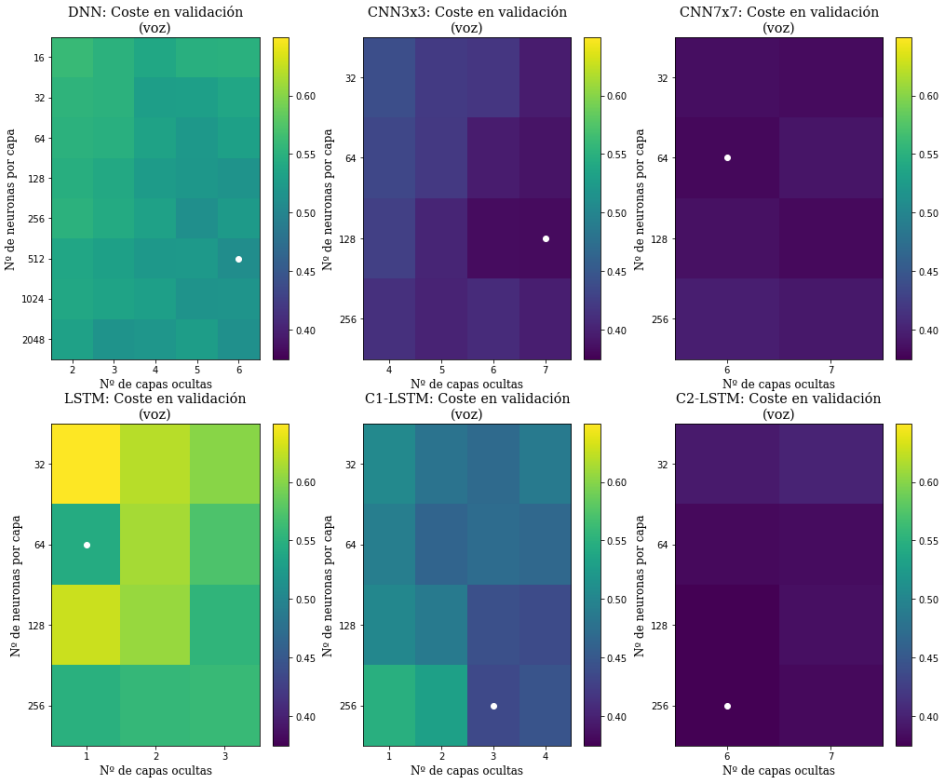

Figura 4.2: Resultados de validación y selección de modelos para detección de voz

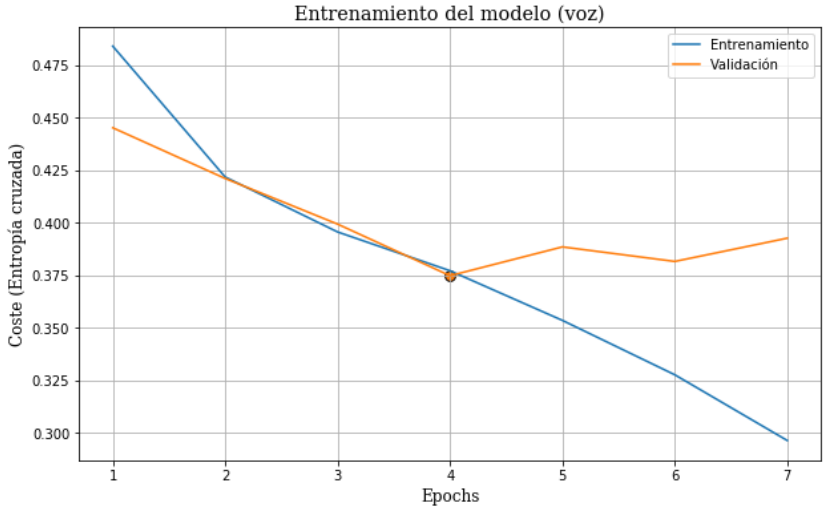

Figura 4.3: Proceso de entrenamiento del mejor modelo para detección de voz

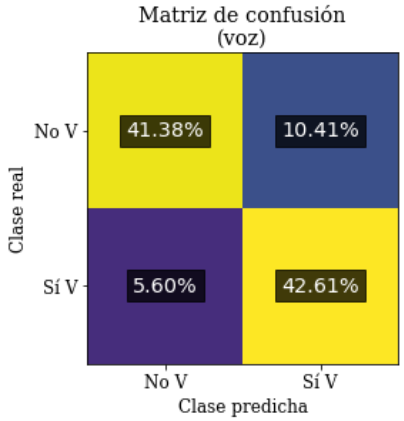

Figura 4.4: Matriz de confusión del mejor modelo para detección de voz sobre el conjunto de test

4.1.2. Resultados de redes neuronales para detección de música

La tabla 4.2 muestra el mejor resultado de cada arquitectura en detección de música. En este caso, el mejor modelo hallado es una **red convolucional** con **6 capas ocultas**, cada una de ellas con **128 filtros** de tamaño $\mathbf{7 \times 7}$. De nuevo, el resultado es **muy similar** entre las redes convolucionales con filtros 3×3 y 7×7 y las C2-LSTM. Esta última arquitectura consigue la mayor tasa de acierto en validación y en test con un número menor de parámetros, pero con una entropía

cruzada ligeramente mayor.

De manera análoga a los resultados de detección de voz descritos en la sección 4.1.1, la figura 4.5 muestra el **coste en validación de los modelos de detección de música** entrenados frente al número de parámetros de cada uno de ellos y la figura 4.6 detalla estos resultados en función del **número de capas** y **número de neuronas por capa**. El entrenamiento del mejor modelo hallado se describe en la figura 4.7, y la matriz de confusión puede verse en la figura 4.8. En este caso, el **9,86%** de los segmentos de test son **falsos positivos** y el **5,94%** son **falsos negativos**.

Modelo	L	N	p	Entrenamiento		Validación		Test	
				Coste	*Acc.*	Coste	*Acc.*	Coste	*Acc.*
DNN	4	2048	7,1504	0,5176	74,73%	0,5515	72,50%	0,5543	72,74%
CNN3x3	7	256	6,5953	0,3618	85,28%	0,3858	84,14%	0,3958	83,51%
CNN7x7	**6**	**128**	**6,6937**	**0,3553**	**85,46%**	**0,3790**	**84,19%**	**0,3792**	**84,20%**
LSTM	3	32	4,5719	0,5592	72,39%	0,5533	72,98%	0,5541	72,65%
C1-LSTM	3	256	6,4034	0,4308	81,08%	0,4658	79,48%	0,4604	79,75%
C2-LSTM	6	128	6,0017	0,3328	86,61%	0,3827	84,34%	0,3799	84,49%

Tabla 4.2: Resultados en distintos modelos para detección de música

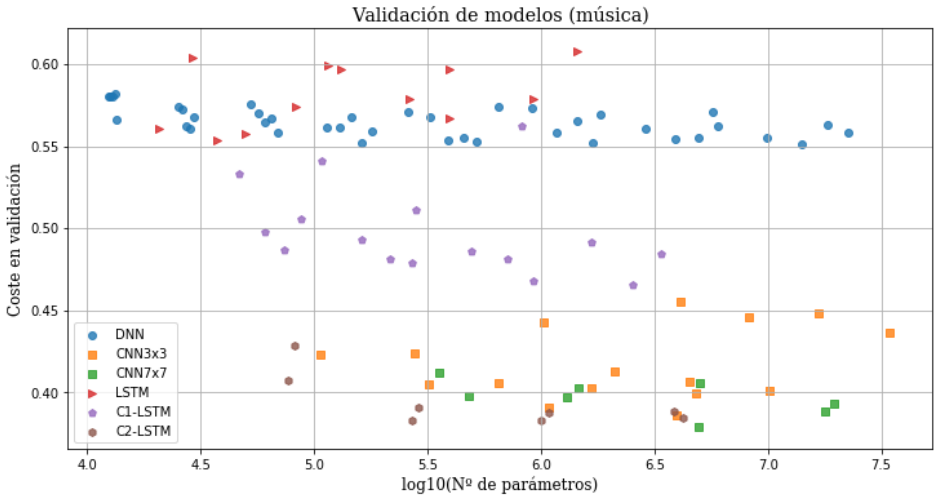

Figura 4.5: Resultados de validación de los modelos para detección de música, frente al número de parámetros

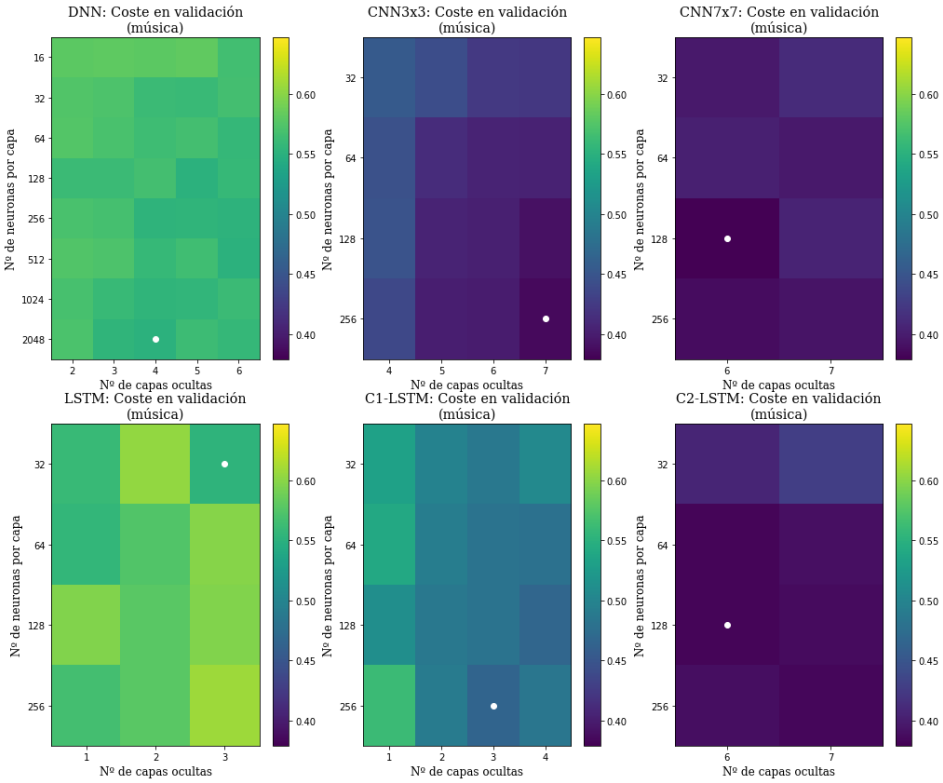

Figura 4.6: Resultados de validación y selección de modelos para detección de música

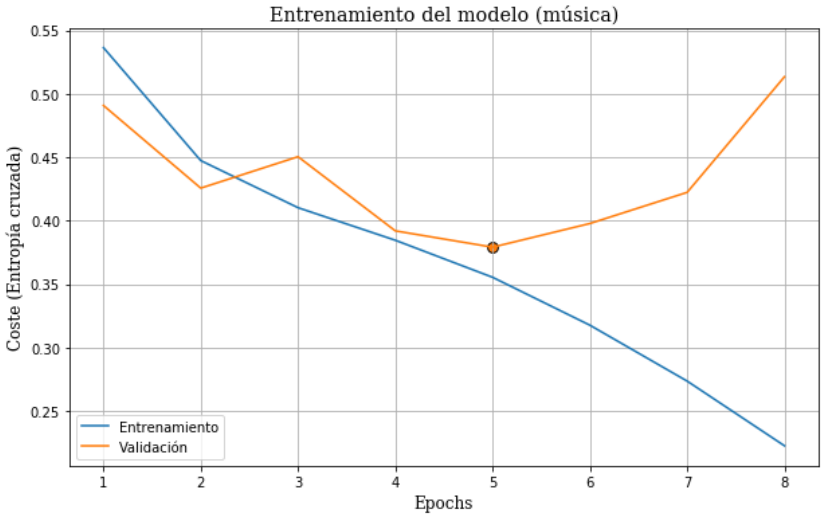

Figura 4.7: Proceso de entrenamiento del mejor modelo para detección de música

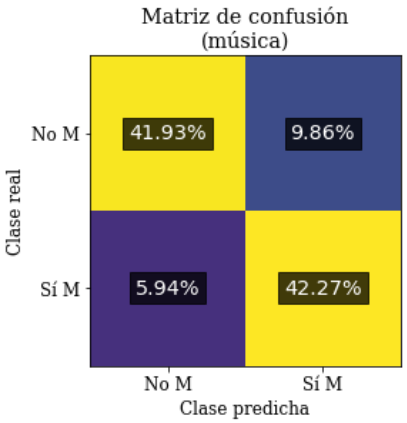

Figura 4.8: Matriz de confusión del mejor modelo para detección de música sobre el conjunto de test

4.1.3. Resultados de redes neuronales para detección conjunta de voz y de música

La tabla 4.3 describe los resultados hallados para la **tarea conjunta de detección de voz y música**. El entrenamiento de estas redes se ha enfocado como un **problema de cuatro clases**. Las **tasas de acierto** mostradas hacen referencia, por tanto, a segmentos en los que se ha predicho correctamente la presencia o ausencia **tanto de voz como de música**. En este caso, se han examinado las **arquitecturas DNN, CNN3x3, C1-LSTM y C2-LSTM**.

Modelo	L	N	p	Entrenamiento		Validación		Test	
				Coste	*Acc.*	Coste	*Acc.*	Coste	*Acc.*
DNN	6	256	5,7696	0,9772	58,93%	1,0376	56,19%	1,0429	55,80%
CNN3x3	6	256	6,6804	0,7260	71,10%	0,7404	70,39%	0,7461	70,37%
C1-LSTM	4	256	6,5298	0,7882	67,58%	0,8769	64,82%	0,8863	64,04%
C2-LSTM	**6**	**256**	**6,5883**	**0,6508**	**74,43%**	**0,7263**	**71,48%**	**0,7328**	**70,98%**

Tabla 4.3: Resultados en distintos modelos para detección conjunta de voz y de música

La mejor configuración hallada es una red con **arquitectura C2-LSTM**, **6 capas convolucionales**, con **256 neuronas** cada una. El **tamaño de batch** utilizado ha sido **8**. El **proceso de entrenamiento** de este modelo se ilustra en la figura 4.11.

Las figuras 4.9 y 4.10 muestran el **coste en validación** de los distintos modelos entrenados. La **entropía cruzada** de los modelos es, en todos los casos,

mayor a la de las redes entrenadas de forma separada para voz o para música: esto es razonable, al tratarse de un problema con más clases.

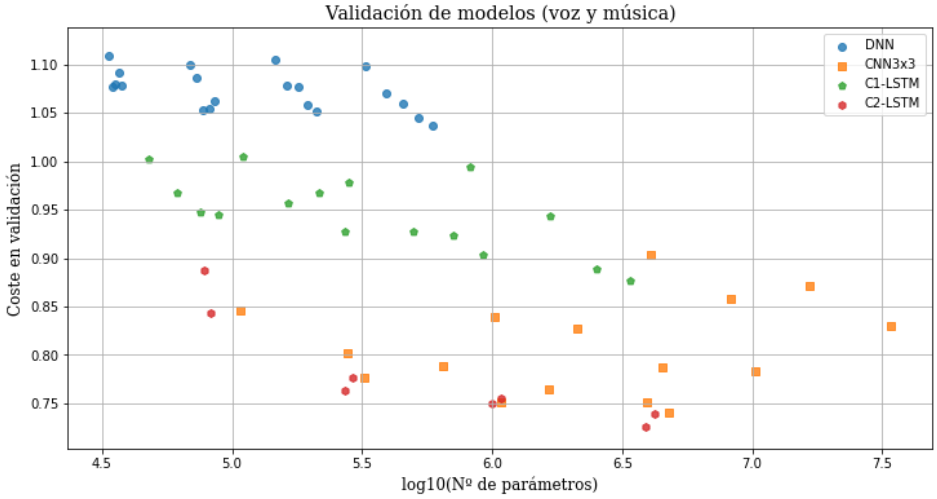

Figura 4.9: Resultados de validación de los modelos para detección conjunta de voz y de música, frente al número de parámetros

La **matriz de confusión** del mejor modelo hallado se muestra en la figura 4.12, junto a la descomposición de la misma en **dos matrices de confusión** correspondientes a la detección de segmentos con **presencia de voz** y segmentos con **presencia de música**. Estas dos matrices, de tamaño 2×2, permiten la comparación con las mostradas en las figuras 4.4 y 4.8.

Con el modelo conjunto, la detección de **voz** en el conjunto de test clasifica un **10,67%** de los segmentos como **falsos positivos** y un **5,52%** como **falsos negativos**. Atendiendo a la etiqueta de **música**, un **8,68%** de los segmentos son **falsos positivos** y un **7,16%** son **falsos negativos**. Las **tasas de acierto** son del **83,81% en detección de voz** y del **84,16% en detección de música**.

4.2. Discusión y análisis de los resultados

Las redes entrenadas para las tareas de detección de voz (4.1.1), detección de música (4.1.2) y detección conjunta de voz y de música (4.1.3) alcanzan resultados **muy similares** en cuanto a **tasas de acierto**, que se recogen en la tabla 4.4. Las tasas de falsos positivos y falsos negativos se expresan como **porcentaje sobre el total de segmentos clasificados**. En ambos casos (música y voz), la precisión es **levemente superior con las redes individuales**, pero no lo suficiente como para considerarlo una diferencia significativa. La **distribución de falsos**

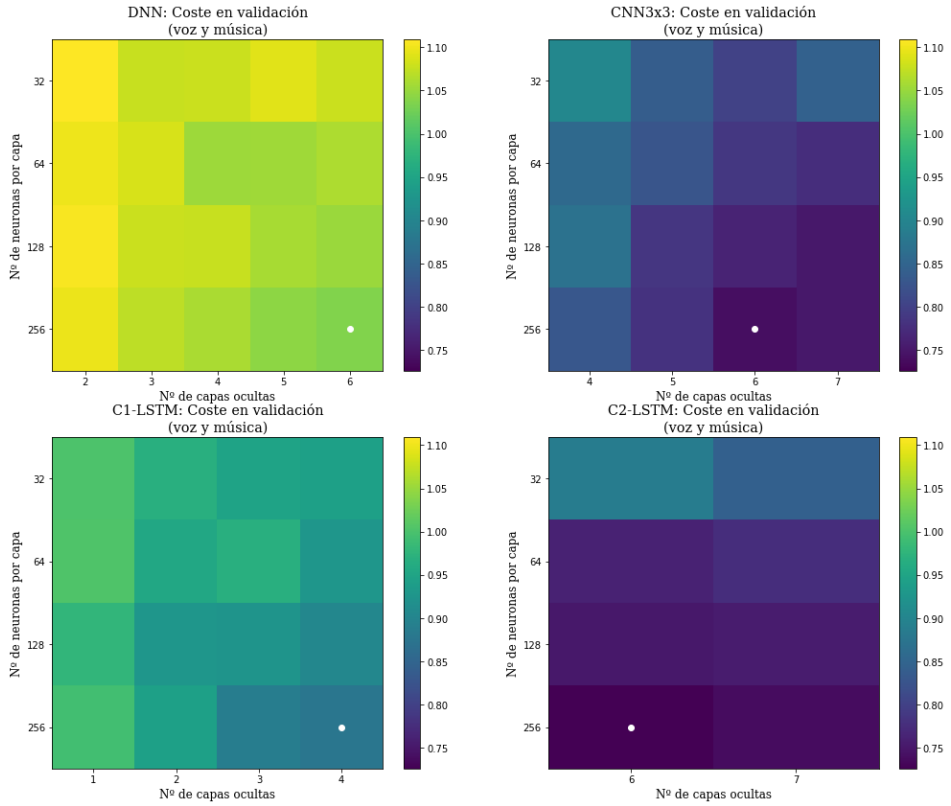

Figura 4.10: Resultados de validación y selección de modelos para detección conjunta de voz y de música

positivos y falsos negativos es también semejante, sobre todo en el caso de la detección de voz.

		Tasa de acierto	Falsos positivos	Falsos negativos
Voz	**Red individual**	**83,99%**	**10,41%**	5,60%
	Red conjunta	83,81%	10,67%	**5,52%**
Música	**Red individual**	**84,20%**	9,86%	**5,94%**
	Red conjunta	84,16%	**8,68%**	7,16%

Tabla 4.4: Resultados obtenidos por las redes individuales y la red conjunta en detección de voz y detección de música

Otra posible comparación entre las redes individuales y la red conjunta es el **número de parámetros entrenables** que contiene cada una de ellas. La red individual seleccionada para **detección de voz** tiene **3.874.050** parámetros, y la red seleccionada para **detección de música** tiene **4.940.162** parámetros. Por otro

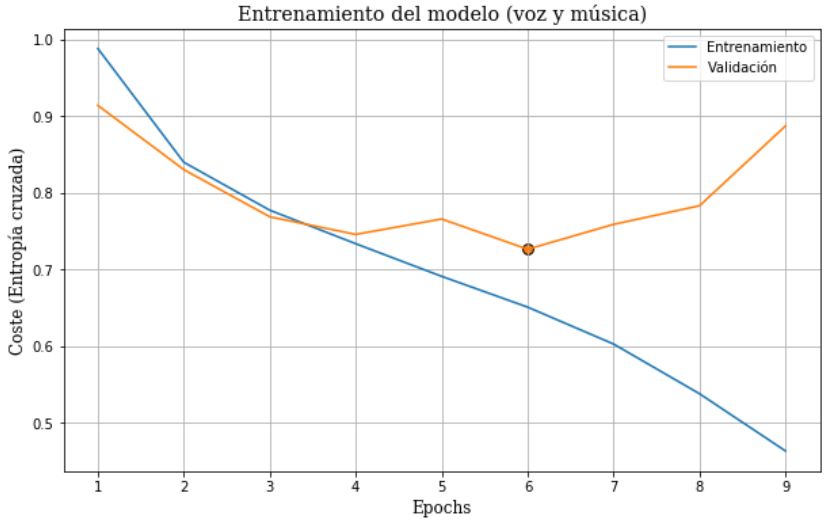

Figura 4.11: Proceso de entrenamiento del mejor modelo para detección conjunta de voz y de música

lado, la **red conjunta** escogida cuenta con **3.875.076** parámetros entrenables, es decir, consigue un **resultado equiparable** al de las dos redes individuales con **menos de la mitad de parámetros** que la suma de ellas.

La **matriz de confusión de cuatro clases** mostrada en la figura 4.12 permite realizar un análisis de la clasificación. La clase con **mayor porcentaje de acierto** es la de los **segmentos con voz y sin música**. Un 20,22% de los segmentos de test pertenecen a esta clase y se clasifican correctamente (es decir, se predice tanto la presencia de voz como la ausencia de música), lo que se corresponde con un **83,87%** de acierto respecto al total de segmentos con voz y sin música. El **80,78%** de los **segmentos con música y sin voz** se clasifican correctamente. Son menores las tasas de acierto para los **segmentos sin voz ni música** (**53,62%**) y los **segmentos con voz y música** (**68,22%**).

Una conclusión derivable de estas tasas de acierto es que la detección de presencia de voz es **menos efectiva** en presencia de música, y viceversa. Además, es recalcable que la clase con **menor porcentaje de acierto** es la de **segmentos que no contienen voz ni música**. Una posible explicación es que en estos segmentos pueden aparecer **otros eventos acústicos** (categorías de AudioSet diferentes a voz y música) que también estén presentes en segmentos con voz y/o música, o sonidos particularmente rítmicos pero sin llegar a ser musicales, llevando a la red a error.

La mayor **variabilidad del rendimiento** se observa entre **diferentes arquitecturas**. Particularmente, las **estructuras CNN**, en especial bidimensionales, son las que obtienen mejores resultados en los tres casos estudiados (voz,

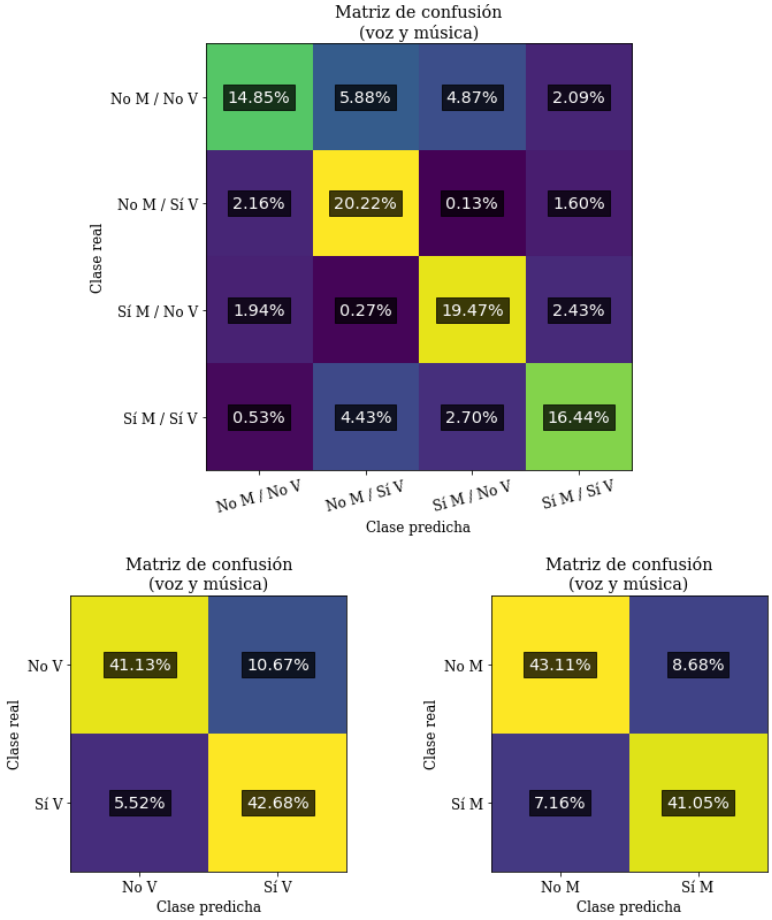

Figura 4.12: Matrices de confusión del mejor modelo para detección conjunta de voz y de música sobre el conjunto de test

música y detección conjunta), frente a las DNN y las LSTM.

Las **estructuras LSTM**, por sí solas, no han logrado un rendimiento mejor que las redes *fully-connected*. Sin embargo, la **etapa convolucional previa**, tanto en el caso unidimensional como el bidimensional, sí permite una **mejora clara frente a las DNN**.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Durante el desarrollo de este **Trabajo Fin de Máster** se ha afrontado una tarea de **reconocimiento de eventos acústicos** sobre un conjunto de **77.936** segmentos de audio de 10 segundos (**216 horas**) pertenecientes a **vídeos de YouTube**, extraídos de la reciente base de datos **AudioSet** de Google. En particular, los eventos estudiados son los correspondientes a **música** y **voz**.

Se han propuesto y comparado **dos enfoques distintos** para el problema: el entrenamiento de **dos redes neuronales separadas**, una para detección de presencia de voz y otra para detección de presencia de música, y el entrenamiento de **una red neuronal conjunta** que se enfrente simultáneamente a ambos retos. Todas estas redes toman como entrada matrices **melgrama** extraídas de los ficheros de audio durante la realización del trabajo.

Han sido evaluadas **diferentes arquitecturas de redes neuronales**, basadas en **DNN**, **CNN** y **LSTM**. Los resultados indican un **rendimiento muy destacable** de las arquitecturas basadas en **redes convolucionales**, que logran rendimientos muy similares en voz y en música, con tasas de acierto en torno al **84%**.

También se ha destacado el **rendimiento de la red conjunta**, que alcanza resultados semejantes a los de las dos redes individuales combinadas, pero contando aproximadamente el **mismo número de parámetros** entrenables que sólo una de ellas.

Además, durante el trabajo se ha diseñado un **entorno experimental** que facilitará el desarrollo de **futuras investigaciones** con la base de datos **Google AudioSet** y con redes neuronales.

5.2. Trabajo futuro

Una de las posibles vías de **continuación de la investigación** iniciada en este trabajo es el estudio de **calibración de los modelos** de clasificación implementados, es decir, la correspondencia entre las probabilidades de pertenencia a cada clase obtenidas a la salida de la red con las probabilidades reales de que el segmento pertenezca a cada una de esas clases.

Otra vía de investigación es la orientada a **ubicar temporalmente los eventos de audio** dentro de cada segmento. El etiquetado ofrecido por Google, sin embargo, no proporciona la información temporal de cada evento, necesaria para afrontar esta tarea desde un enfoque supervisado.

Por último, un análisis similar al llevado a cabo durante este trabajo podría ser aplicado a **otras clases de eventos** pertenecientes a AudioSet y a **modelos de clasificación diferentes**, como los **modelos de atención** [9].

Bibliografía

[1] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, (New Orleans, LA), 2017.

[2] D. Castán, D. Tavarez, P. Lopez-Otero, J. Franco-Pedroso, H. Delgado, E. Navas, L. Docio-Fernández, D. Ramos, J. Serrano, A. Ortega, and E. Lleida, "Albayzín-2014 evaluation: audio segmentation and classification in broadcast news domains," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, p. 33, Dec 2015.

[3] B. G. Naranjo and J. González-Rodríguez, "Segmentación de audio broadcast," Trabajo Fin de Grado, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Jan 2016.

[4] Y. Zhu, M. S. Kankanhalli, and S. Gao, "Music key detection for musical audio," in *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International*, pp. 30–37, IEEE, 2005.

[5] J. G. Fiscus, J. Ajot, M. Michet, and J. S. Garofolo, "The rich transcription 2006 spring meeting recognition evaluation," *NIST 2006 Spring Rich Transcrition Evaluation Workshop*, 2006.

[6] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "CNN architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 131–135, IEEE, 2017.

[7] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," *CoRR*, vol. abs/1710.00343, 2017.

[8] Y. Wu and T. Lee, "Reducing Model Complexity for DNN Based Large-Scale Audio Classification," *CoRR*, vol. abs/1711.00229, 2017.

[9] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, "Audio Set classification with attention model: A probabilistic perspective," *CoRR*, vol. abs/1711.00927, 2017.

[10] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and Classification of Acoustic Scenes and Events," *IEEE Transactions on Multimedia*, vol. 17, pp. 1733–1746, Oct 2015.

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, Sep 1995.

[12] A. Varga and R. Moore, "Hidden Markov Model decomposition of speech and noise," in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pp. 845–848, IEEE, 1990.

[13] "Open Speech Analytic Technologies Pilot Evaluation OpenSAT Pilot Evaluation Plan: version 1.1," 2017.

[14] D. de Benito Gorrón and J. González-Rodríguez, "Detección de música en contenidos multimedia mediante ritmo y armonía," Trabajo Fin de Grado, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Jun 2017.

[15] R. Zazo, A. Lozano-Diez, J. Gonzalez-Dominguez, D. T. Toledano, and J. Gonzalez-Rodriguez, "Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks," *PloS one*, January 2016.

[16] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker diarization: A review of recent research," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 356–370, Feb 2012.

[17] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications and challenges," *IEEE Signal Processing Magazine*, In Press (2013).

[18] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[19] Álvaro Escudero-Barrero, A. Lozano-Diez, R. Zazo, J. Franco-Pedroso, D. T. Toledano, and J. González-Rodríguez, "Speech activity detection combining DNN-trained and rule-based voice and music detectors at NIST OpenSAT 2017." Pendiente de publicación, 2017.

[20] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[21] J. L. Elman, "Finding structure in time," *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990.

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Trans. Neur. Netw.*, vol. 5, pp. 157–166, Mar. 1994.

[23] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR*, vol. abs/1211.5063, 2012.

[24] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1* (D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, eds.), pp. 318–362, Cambridge, MA, USA: MIT Press, 1986.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, vol. 60, pp. 84–90, May 2017.

[29] A. Temko, R. Malkin, C. Zieger, D. Macho, C. Nadeu, and M. Omologo, "CLEAR Evaluation of Acoustic Event Detection and Classification Systems," in *Proceedings of the 1st International Evaluation Conference on Classification of Events, Activities and Relationships*, CLEAR'06, (Berlin, Heidelberg), pp. 311–322, Springer-Verlag, 2007.

[30] A. Singhal, "Introducing the Knowledge Graph: Things, not strings," *Official Google blog*, 2012.

[31] B. McFee, M. McVicar, C. Raffel, D. Liang, O. Nieto, J. Moore, D. Ellis, D. Repetto, P. Viktorin, J. F. Santos, and A. Holovaty, "librosa: v0.4.0," June 2015.

[32] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[33] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," in *OSDI*, vol. 16, pp. 265–283, 2016.

[34] L. Prechelt, "Early stopping—but when?," in *Neural networks: tricks of the trade*, pp. 53–67, Springer, 2012.

[35] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670, ACM, 2014.

[36] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *CoRR*, vol. abs/1206.5533, 2012.