UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Modelos de atención en redes neuronales profundas para detección de eventos de audio

Guillermo Segovia Fernández
Tutor: Doroteo Torre Toledano

Junio 2020

Modelos de atención en redes neuronales profundas para detección de eventos de audio

AUTOR: Guillermo Segovia Fernández
TUTOR: Doroteo Torre Toledano

Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2020

Resumen

Este Trabajo de Fin de Grado trata de estudiar los posibles beneficios que se pueden obtener de la utilización de mecanismos de atención en los problemas de clasificación y detección de eventos de audio. Este problema trata sobre localizar los intervalos de la grabación donde ocurren los eventos de audio, además de clasificarlos en las distintas clases del problema. En concreto se quiere estudiar si es posible, a través de la información que se obtiene de los mecanismos de atención, localizar los eventos de audio dentro de una grabación sin haber entrenado la red neuronal con marcas temporales, es decir, entrenando la red con etiquetas a nivel de grabación.

El trabajo se realizará dentro del marco de la evaluación *Challenge 2020 Task 4* de DCASE, junto con el grupo de investigación de AUDIAS de la Universidad Autónoma de Madrid. Es por esto que el conjunto de datos utilizados tanto para el entrenamiento como para la evaluación de este trabajo son un subconjunto del proporcionado para este concurso. Además, el modelo de partida sobre el que se introducirán cambios para este trabajo pertenece a los ganadores de la convocatoria de 2019 de esta misma evaluación.

El conjunto de datos utilizado tiene grabaciones con dos tipos de etiquetado. Por un lado, hay grabaciones etiquetadas a nivel de archivo, es decir, tan solo se dice que clases, o eventos de audio, se dan en la grabación. Por el otro lado, hay grabaciones etiquetadas con marcas temporales, es decir, además de decir las clases que se dan en la grabación, se indica el intervalo temporal en el que se dan.

Con este objetivo en mente, se preparará un modelo típico de clasificación de audios sin localización de eventos. Este modelo sigue la línea del estado del arte al estar compuesto por un bloque convolucional seguido de uno recurrente. A este modelo se le añadirá, como una parte del mismo, un mecanismo de atención que más adelante será objeto de estudio. El objetivo es observar si, a través de un entrenamiento no dirigido a la localización temporal de eventos, la red neuronal y, en concreto, el mecanismo de atención, son capaces de obtener información sobre los intervalos en los que ocurren los eventos.

Tras varias iteraciones sobre distintos modelos, el objetivo se acaba logrando. Una vez entrenada la red, se dispone el mecanismo de atención como capa de salida para analizarla. Se observa que, en los intervalos en los que en el espectrograma se marca la ocurrencia del evento del audio, también se da un aumento en el valor de salida del mecanismo de atención. A pesar de este logro, se encuentran algunas dificultades. Por ejemplo, el modelo de atención no es capaz de separar entre las distintas clases. Los distintos vectores de atención para cada una de las clases resaltan la ocurrencia del resto de clases, por lo que en una grabación donde se da más de una clase, no queda claro que activaciones hacen referencia a una clase y cual a la otra. Lo que sí queda claro es el potencial de estos mecanismos y del mundo de posibilidades que se presenta.

Abstract

This Final Degree Project tries to study the possible benefits that can be obtained from the use of attention mechanisms in the problems of classification and detection of audio events. This problem is about locating the intervals in the recording where the audio events occur, as well as classifying them into the different classes of the problem. Specifically, we want to study if it is possible, through the information obtained from the attention mechanisms, to locate the audio events within a recording without having trained the neural network with time stamps, that is, training the network with labels at the recording level.

The work will be carried out within the context of the DCASE Challenge 2020 Task 4 evaluation, together with the AUDIAS research group of the Universidad Autónoma de Madrid. This is why the dataset used for both training and evaluation of this work is a subset of that provided for this contest. In addition, the starting model on which changes will be made for this work belongs to the winners of the 2019 call for this same evaluation.

The dataset used has recordings with two types of labeling. On the one hand, there are recordings tagged at the file level, that is, only classes, or audio events, are said to occur in the recording. On the other hand, there are recordings tagged with time stamps, that is, in addition to saying the classes given in the recording, the time interval in which they are given is indicated.

With this objective in mind, a typical audio classification model without event localization will be prepared. This model follows the line of the state of the art as it is made up of a convolutional block followed by a recurring one. A care mechanism will be added to this model, as part of it, which will be studied later. The objective is to observe whether, through training not directed at the temporal location of events, the neural network and, specifically, the attention mechanism, are able to obtain information about the intervals at which events occur, so that, after due processing, obtain the time stamps.

After several iterations on different models, the objective is finally achieved. Once the network is trained, the attention mechanism is arranged as an output layer to analyze it. It is observed that, in the intervals in which the occurrence of the audio event is marked on the spectrogram, there is also an increase in the output value of the attention mechanism. Despite this achievement, some difficulties are encountered. For example, the care model is not capable of separating between the different classes. The different attention vectors for each of the classes highlight the occurrence of the other classes, so in a recording where more than one class is given, it is not clear which activations refer to one class and which to the other. What is clear is the potential of these mechanisms and the world of possibilities that arises.

Palabras clave

Detección de eventos de audio, clasificación de eventos de audio, mecanismos de atención en redes neuronales, CNN, RNN, CRNN, LSTM, GRU, Python, libROSA, DCASE

Keywords

Audio events detection, audio events classification, attention models in neural networks, CNN, RNN, CRNN, LSTM, GRU, Python, libROSA, DCASE

Agradecimientos

A mi familia por aguantarme
y al equipo de AUDIAS por haberme
acompañado en esta odisea

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Esta memoria de TFG trata sobre la aplicación de modelos o mecanismos de atención en localización de eventos de audio. Estos mecanismos de atención son ampliamente usados en muchos campos del mundo de la inteligencia artificial tales como el procesamiento del lenguaje natural, pero sus posibilidades son aún un mundo por explorar.

Esta investigación se hará a través de la participación, junto con el equipo de AUDIAS de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid, en el *Challenge 2020 Task 4* de la comunidad DCASE. Esta tarea consiste concretamente en la clasificación y localización de eventos de audio.

Entrando en más detalle, la tarea de clasificar y localizar eventos de audio consiste en determinar que clases se dan dentro de una grabación y en que intervalos de tiempo ocurren éstas. De esta forma, si se quiere que la red aprenda a localizar eventos, el etiquetado se adapta incluyendo las marcas temporales de cada evento. El objetivo de este trabajo es determinar si, a través de los mecanismos de atención, es posible obtener la localización de los eventos entrenando con etiquetado débil, es decir, que las etiquetas solo indiquen las clases que se dan en la grabación, sin marcas temporales.

La red neuronal planteada para el trabajo deriva de la ganadora que se presentó el año anterior para la misma tarea de DCASE, aunque ésta no incluía ningún mecanismo de atención. La red ganadora de 2019 sigue la tendencia habitual de los últimos años, siendo una red compuesta por un primer bloque convolucional, que analizará los espectrogramas de entrada, seguido de un bloque recurrente que integrará la información obtenida durante toda la secuencia temporal y dará las predicciones de salida.

En general, se ha demostrado que este tipo de redes neuronales funcionan bien a la hora de clasificar grabaciones en las distintas clases de un problema, obviando toda la parte de localización de eventos. Es por esto que se pretende partir de una buena base de clasificación para aproximarnos a la localización de eventos.

A lo largo de este trabajo se ha incluido el mecanismo de atención dentro de la red neuronal propuesta para analizar los efectos que este provoca en su rendimiento y, mas adelante, analizar la salida que genera. En el análisis de la salida lo que se pretende es comprobar si el mecanismo de atención ha aprendido de manera independiente a localizar los eventos de audio dentro de las grabaciones.

1.2 Objetivos

Con este trabajo se busca estudiar las posibilidades que ofrecen los mecanismos de atención dentro del ámbito de la clasificación y localización de eventos de audio. Se quiere comprobar si los mecanismos de atención otorgan a una red neuronal, entrenada sólo para determinar la ocurrencia o no de una clase, la capacidad de localizar eventos en la grabación, a pesar de que el entrenamiento no este dirigido a localizar los eventos. Las grabaciones con las que se entrenará la red tan solo estarán etiquetadas con las clases que incluyen y no con marcas temporales que localicen los eventos.

Una vez entrenada la red neuronal, se estudiará la salida que genera el mecanismo de atención ante una nueva grabación y se verá si, por su naturaleza, ha sido capaz de enfocarse y resaltar los momentos en los que ocurre el evento.

Para lograrlo se adaptará un modelo típico de clasificación de audio, en concreto el ganador del *Challenge 2019 Task 4* de DCASE, y se entrenará tan solo con etiquetas débiles, es decir, sin marcas temporales que localicen los eventos de audio. Esto ayudará a determinar si el mecanismo de atención es capaz de enfocarse en los eventos de audio y de esta forma enfatizarlos y destacarlos en el ejemplo de entrada. De esta forma, tras el análisis oportuno de la salida del mecanismo de atención, se podrán determinar las marcas temporales de los eventos de audio a través de un entrenamiento sin éstas.

1.3 Organización de la memoria

Esta memoria consta de un total de 6 secciones. A lo largo de las distintas secciones la memoria trata de explicar los pasos seguidos en este trabajo para alcanzar el objetivo final. El documento cuenta con las siguientes secciones:

- **Introducción**: en esta sección se presentan las motivaciones y los objetivos básicos en los que se ha basado este trabajo.

- **Estado del arte**: en esta sección se presenta el problema al que se va hacer frente durante el trabajo, así como la participación en la evaluación de DCASE *Challenge 2020 Task 4* junto con el equipo de AUDIAS. Además, se explican los fundamentos básicos de las redes típicas a la hora de afrontar el problema planteado.

- **Dataset**: en esta sección se explica la importancia de la elección de un buen conjunto de datos para este problema. También se detalla la búsqueda del conjunto de datos y el porqué de la elección del dataset definitivo. Por último, se realiza un estudio del dataset que será útil para el análisis de resultados.

- **Desarrollo**: en esta sección se trata todo el proceso de creación del modelo, desde el preprocesamiento de la entrada hasta los parámetros de entrenamiento, pasando por la creación del modelo de atención.

- **Análisis de los resultados**: en esta sección se hace una comparativa entre distintos modelos planteados para ver las mejoras que aportan los distintos cambios. Por último, se analiza la posibilidad de localizar eventos con ayuda de la salida del mecanismo de atención.

- **Conclusiones y trabajo futuro**: como sección final, en esta se habla de las conclusiones finales del trabajo, así como de los posibles caminos que se abren para la investigación en el futuro.

2 Estado del arte

2.1 Problema de detección y clasificación de eventos de audio

El problema de detección y clasificación de eventos de audio no solo consiste en determinar qué clases se dan dentro de una grabación. Además de esto, la solución tiene que ser capaz de determinar las marcas temporales en las que se da el evento dentro del audio. Es uno de los principales problemas a la hora de hablar de aprendizaje automático y audio. Ha ido ganando importancia en los últimos años hasta colocarse en el punto de mira de muchos investigadores.

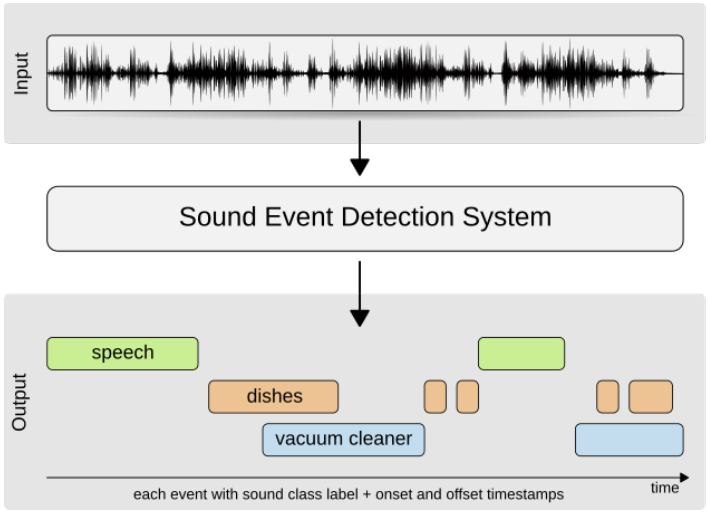

Figura 1-Detección eventos de audio

El problema no solo consiste en determinar qué eventos ocurren en un audio en concreto. Lo que realmente se pretende es identificar en que intervalos, dentro del audio, ocurre cada evento.

De esta intención derivan los siguientes problemas. En una grabación de audio típicamente se tiene más de un evento, lo que lo convierte directamente en un problema *multi-label*. Además, estos eventos de audio pueden solaparse en el tiempo, lo que significa que un evento puede terminar o empezar en mitad de otro evento. Si a todo esto se le añaden los problemas típicos del audio, como, por ejemplo, *datasets* mal etiquetados o con ruido, hacen que el problema se complique aun más.

2.2 Tipos de redes neuronales

En esta sección se presentarán los tipos de redes que se han usado para resolver el problema planteado en este trabajo.

2.2.1 Redes neuronales convolucionales

Los orígenes de las redes neuronales convolucionales se encuentran en el Neocognitron de Kunihiko Fukushima [1] publicado en 1980. Aunque se siguió avanzando en este tipo de redes durante muchos años, no se consiguió sacar provecho por el coste computacional. No fue hasta 2012 cuando se dio el gran salto respecto a las redes convolucionales con AlexNet

[2]. Gracias a las implementaciones con GPUs, se pudo dar mas profundidad a la red convolucional, lo que aumento en gran medida su rendimiento, ganando así el *ImageNet Large Scale Visual Recognition Challenge* en 2012.
Las redes convolucionales surgen a raíz de una falta de rendimiento para determinadas tareas por parte de las típicas capas densas o perceptrones. La arquitectura de estas redes es análoga a los patrones de neuronas que se observan en el córtex visual del cerebro humano. Las redes convolucionales ofrecen algunas ventajas sobre las densas en algunos aspectos muy importantes para algunos problemas. Por ejemplo, las redes convolucionales aplican sus pesos a través de matrices bidimensionales, por lo que la entrada de estas puede ser directamente una matriz de datos sin la necesidad de ser convertida a un vector. De esta forma, este tipo de redes consigue poner en relación un valor de la matriz con los que le rodean. Además, el número de parámetros de estas redes es sustancialmente menor al que se tendría al intentar procesar una imagen con una capa completamente conectada. Es por esto que son muy útiles para la clasificación de imágenes.

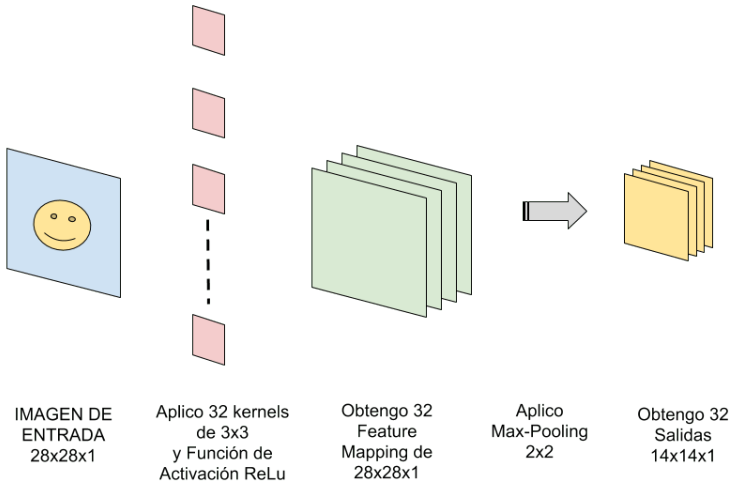

Figura 2-Ejemplo etapa convolucional

Normalmente las redes convolucionales tienen tres componentes básicos. Como primer componente y el más necesario de todos, se tienen las capas de filtro convolucional. A través del entrenamiento de *kernels*, estas capas consiguen extraer de la matriz de datos de entrada las características más importantes para la tarea de clasificación. Estos *kernels* se irán deslizando y aplicando sobre toda la matriz de datos de entrada. De esta forma se generan tantas matrices como *kernels* tenga la capa convolucional, cada una de estas matrices filtrada por un *kernel*. Como segundo elemento y para otorgar cierto rango de tolerancia, se añaden capas de reducción de dimensionalidad. Hay muchos tipos de capas que reducen la dimensionalidad dependiendo de la operación que utilicen para fusionar las distintas celdas. Algunos ejemplos son *Average pooling* o *Max pooling*. Estas capas van reduciendo distintas regiones de la matriz de datos haciendo la media o quedándose con el valor máximo de la región. Por último, después de haberse extraído las características, hace falta una capa de clasificación final para determinar de qué clase es el ejemplo estudiado.

2.2.2 Redes neuronales recurrentes

Al igual que las redes convolucionales, las recurrentes también surgieron en la década de los 80. De la misma forma, tampoco se les pudo sacar provecho por el alto coste computacional hasta estos últimos años, cuando se ha avanzado increíblemente en el ámbito del hardware y se ha popularizado la industria del aprendizaje automático.

El aspecto innovador de este tipo de redes reside en la realimentación de las neuronas que la componen. Hasta el momento solo se habían visto neuronas que, a raíz de una entrada, generaban una salida que propagaban únicamente hacia delante. Las redes recurrentes proponen neuronas que reutilizan la salida generada, teniendo entonces dos entradas. De esta forma consiguen crear temporalidad, tener en cuenta la dimensión del tiempo y por tanto tener memoria. Es por esto que son las redes preferidas para trabajar con series temporales o con lenguaje natural [3].

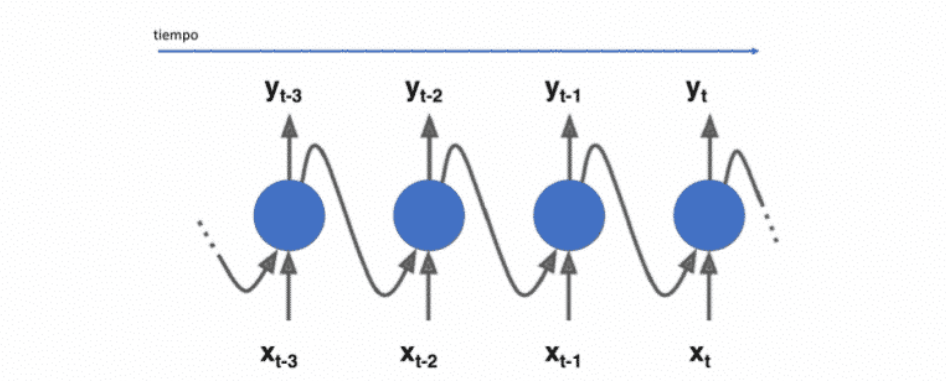

Figura 3-Ejemplo básico red recurrente

En la *Figura 3* se ejemplifica el funcionamiento de una red recurrente básica. Dependiendo de la complejidad de las propias neuronas y de la información que se transmite de un punto temporal al siguiente surgen las distintas redes neuronales recurrentes. Las mas importantes a día de hoy son las redes LSTM y las redes GRU. Las redes LSTM surgen en 1997 solucionando problemas importantes de las redes recurrentes como el desvanecimiento del gradiente, que dificultaba el aprendizaje profundo, al tener en cuenta el error presente y arrastrar el error pasado. Las GRU surgen en 2014, son una simplificación de las LSTM y demuestran un mejor rendimiento en conjuntos de datos mas pequeños [4].

2.2.3 Modelos de atención

Los modelos o mecanismos de atención surgen hace relativamente poco tiempo. En estos últimos años han ganado popularidad en determinadas tareas del mundo del aprendizaje automático como la traducción automática y otras tareas de procesamiento de lenguaje natural.

Los modelos de atención hacen referencia a métodos de procesamiento de entrada de las redes neuronales. Al principio de su desarrollo formaban parte del conjunto de piezas que conforma una red neuronal. Su propósito consistía en ser capaces de resaltar los aspectos mas importantes de la entrada de la red neuronal, facilitando así la tarea de reconocimiento al resto de la red neuronal. En los últimos estudios también se ha podido comprobar el buen

funcionamiento de los modelos de atención clasificando de forma independiente, como el caso del *Transformer* [5].

A fin de cuentas, los modelos de atención permiten que las redes neuronales se comporten como el sistema de enfoque visual del ser humano. De la misma forma en la que un ser humano cuando observa una imagen es capaz de enfocar a distintas zonas de la escena, el modelo de atención permite a la red neuronal "enfocar" distintas partes de la entrada. Según la red va aprendiendo a clasificar, el modelo de atención irá variando su enfoque para encontrar qué zonas son las mas importantes para la tarea de clasificación.

El campo donde más impacto tienen y donde más se están usando es, como se ha mencionado anteriormente, el campo del procesamiento del lenguaje. Sin embargo, investigadores de otros campos han visto su potencial y se realizan muchos estudios para ver el impacto y mejoras que pueden tener en otros campos. Por ejemplo, en el campo de reconocimiento de voz y en el de clasificación de imágenes también ha supuesto un gran avance al dar a conocer en qué se esta fijando realmente la red neuronal para clasificar. Ese también es el objetivo de este trabajo.

2.3 Evaluaciones DCASE

Para entrar en contexto, antes se explicará que es DCASE [6]. DCASE es una plataforma creada por investigadores del área de la detección y clasificación de sonidos y audios. Surge en el año 2013 con la intención de crear una comunidad en la que incluir a investigadores de todas las partes del mundo y unir fuerzas para avanzar en el mundo del aprendizaje automático y clasificación de audios. Para lograr este objetivo, además de organizar charlas y convenciones varias veces al año a lo largo del mundo, crean anualmente una serie de evaluaciones en distintos ámbitos de la clasificación de audios, enfrentando así a cientos de grupos de investigación a través de internet para buscar una mejor solución a los principales problemas del mundo de los audios. Cada año las evaluaciones cambian en algún aspecto para incentivar así la investigación.

El grupo de investigación de AUDIAS de la Escuela Politécnica Superior se presenta este año a la evaluación de una de las tareas. En concreto a la *Task 4* del *Challenge 2020*, que se llama *Sound Event Detection and Separation in Domestic Environments*. Dado que la tarea en años anteriores tenía bastante que ver con el propósito de este trabajo de fin de grado y que era una buena oportunidad, entré a formar parte del grupo de AUDIAS que iba a competir en la evaluación de esta tarea.

3 Dataset

En esta sección se tratarán los problemas relativos a la elección del dataset, así como la explicación y análisis del dataset finalmente escogido para la realización de este trabajo.

3.1 Elección de dataset

La elección del dataset es de crucial importancia para cualquier trabajo de aprendizaje automático. Escoger un mal dataset puede acarrear muchos problemas en el desarrollo del proyecto además de un posible peor resultado. Es por esto que, a la hora de escogerlo para este trabajo de fin de grado, se han tomado las medidas de precaución necesarias y se ha estudiado más de una posibilidad. Se necesitaba algo bastante concreto y que se adaptase a las necesidades de este trabajo, es decir, que tuviera tanto etiquetas débiles como etiquetas fuertes y un grado alto de fiabilidad. El etiquetado débil consiste en indicar que clases se dan en cada ejemplo, mientras que el etiquetado fuerte, además de esto último, indica los intervalos en los que se dan esos eventos de audio. Sobre este tema se profundiza más adelante en esta sección.

3.1.1 ESC-50

En primera instancia y para entrar en contacto con el mundo de los audios, se hicieron algunos experimentos con el dataset ESC-50 [7]. Estos experimentos consistieron en extracción de características de los audios con ayuda de librerías de Python como libROSA, tema que se trata mas adelante en la sección 4.1, y pruebas con CRNN [8] simples para la clasificación.

Este dataset consiste en un conjunto de 2.000 grabaciones de sonido ambiente. Estos audios se reparten en 50 clases, es decir, 40 audios por clase. Los audios tienen una duración de 5 segundos y están grabados a 44,1 kHz, por un solo canal y comprimidos con Ogg Vorbis a 192Kbps.

Las etiquetas de este dataset no se han revisado a mano, sino que mantiene las etiquetas del usuario que publicó la grabación. Además, estas etiquetas son etiquetas débiles, es decir, solo indican si la clase ocurre o no en el audio, no los tiempos en los que ocurre. Por último, 40 ejemplos por clase son relativamente pocos para el trabajo que se quiere realizar. Es por todo esto que no se optó por seguir con este dataset y seguir buscando otro que se adaptase mejor a las necesidades del trabajo.

3.1.2 DCASE 2020 Tarea 4

Como se comenta en secciones anteriores, dadas las características de esta evaluación, el dataset de esta evaluación es el idóneo para llevar a cabo este proyecto. Para este trabajo no se ha utilizado la totalidad del dataset que se proporcionaba para esta tarea, pues la tarea incluye una parte de separación de fuentes de audio que no se trata en este trabajo. En concreto se utilizará parte del dataset DESED [9] creado por investigadores de DCASE. Este

es un subconjunto de audios de Google Audio Set [10], el cual se compone de alrededor de 2 millones de segmentos de 10 segundos extraídos de videos de YouTube.

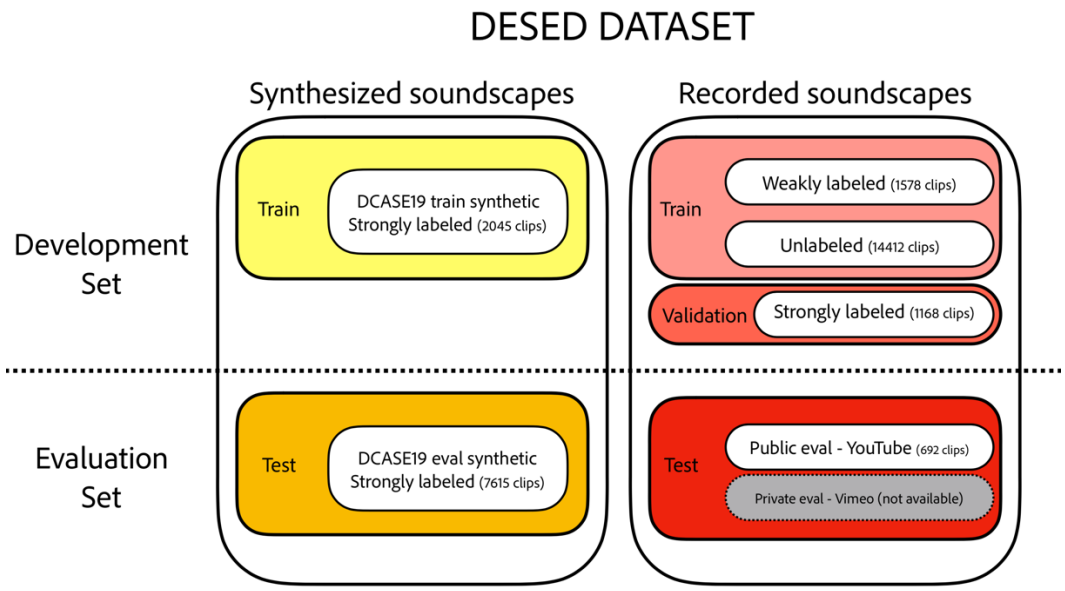

Figura 4-Componentes DESED

DESED es un dataset que sigue en desarrollo y va creciendo año a año. Actualmente se compone de grabaciones reales y audios sintéticos, como se ve en la *Figura 4*.

Las grabaciones reales tienen tanto conjuntos con etiquetas débiles como con etiquetas fuertes y también un conjunto sin etiquetar con muchos mas ejemplos que los otros. Lo especial de este dataset y que resuelve el principal problema de Google Audio Set es que las etiquetas están revisadas a mano, es por esto que el número de ejemplos es mucho menor que el original. En la evaluación del año 2019 se incluyeron los audios sintéticos, perfectamente etiquetados, pero no tan naturales como las grabaciones, con el propósito de ver su impacto en los entrenamientos. Para este trabajo nos quedaremos con los conjuntos de la *Figura 5*, que son, el de entrenamiento con etiquetas débiles, el de validación con etiquetas fuertes y el de evaluación pública con etiquetas fuertes.

Dataset	Subset	Type	Usage	Annotations	Sampling frequency
DESED	Real: weakly labeled	Real soundscapes	Training	Weak labels (no timestamps)	44.1kHz
	Real: validation	Real soundscapes	Validation	Strong labels (with timestamps)	44.1kHz
	Real: public evaluation	Real soundscapes	Evaluation **(do not use this subset to tune hyperparamters)**	Strong labels (with timestamps)	44.1kHz

Figura 5-Dataset usado de DESED

3.2 Clases de eventos acústicos

Como se comenta en los párrafos anteriores, lo que se usará de DESED es un subconjunto de unos 3.000 audios de Google Audio Set, esto es tan solo un 0,15% de su tamaño, por lo que el número de clases de los dataset escogidos no podrá ser tan variada. Google Audio Set tiene 527 clases diferentes y si tuviéramos esas mismas clases con tan pocos ejemplos, apenas se tendrían 5 ejemplos por clase. Es por esto que las clases de DESED son diez como se ve en la *Figura 6*.

- Speech `Speech`
- Dog `Dog`
- Cat `Cat`
- Alarm/bell/ringing `Alarm_bell_ringing`
- Dishes `Dishes`
- Frying `Frying`
- Blender `Blender`
- Running water `Running_water`
- Vacuum cleaner `Vacuum_cleaner`
- Electric shaver/toothbrush `Electric_shaver_toothbrush`

Figura 6-Clases DESED

Las etiquetas del dataset pueden ser débiles o fuertes:

- **Etiquetas débiles**: que una etiqueta sea débil significa que esta indica simplemente si la clase ocurre o no en el audio. El conjunto de etiquetas débiles viene descrito en un fichero csv de la siguiente forma:

filename	*event_labels*
YKK227gPpRn4_30.000_40.000.wav	*Alarm_bell_ringing,Speech*

  La primera columna, *filename*, hace referencia al nombre del fichero de audio. La primera parte del nombre es el código de identificación del vídeo en YouTube, en este caso *YKK227gPpRn4*. La segunda es el segundo de inicio del audio dentro del video de YouTube, en este caso el segundo 30. La tercera es el segundo de fin del audio dentro del video de YouTube, en este caso 40. Como se ve, los audios duran 10 segundos.

  En la segunda columna, *event_labels*, encontramos las etiquetas débiles separadas por comas. Cada una de las etiquetas de las clases que se dan en el audio quedan reflejadas en esta columna.

- **Etiquetas fuertes**: que una etiqueta sea fuerte significa que no solo indica que el evento ocurre en el audio, sino que además da las marcas de tiempos en los que este ocurre. Los conjuntos con etiquetado fuerte también vienen descritos en ficheros csv de la siguiente forma:

filename	*onset*	*offset*	*event_label*
G5wlhYFdlxw_508_518.wav	*0.000*	*10.000*	*Running_water*
G5wlhYFdlxw_508_518.wav	*3.448*	*3.698*	*Dishes*
G5wlhYFdlxw_508_518.wav	*4.650*	*6.873*	*Speech*
G5wlhYFdlxw_508_518.wav	*5.306*	*5.758*	*Dishes*

En este caso se tienen cuatro en vez de dos columnas. La primera columna, *filename*, se codifica de la misma forma que en el etiquetado débil.

La segunda columna, *onset*, indica el instante de tiempo en el que empieza el evento de audio dentro de los 10 segundos de duración que tiene.

La tercera columna, *offset*, indica el instante de tiempo en el que acaba el evento de audio.

Por último, la columna *event_label* indica la clase que se da en ese lapso de tiempo.

3.3 Análisis del dataset

El dataset se compone finalmente de tres conjuntos:

- **Weakly labeled**: este dataset tiene un total de 1578 audios con etiquetas debiles.
- **Validation**: conjunto con etiquetas fuertes y un total de 1153 audios. Según los creadores de DESED son 1168, pero hay algunos que no se han podido recuperar.
- **Public evaluation**: un total de 692 audios con etiquetado fuerte.

Para el entrenamiento de este trabajo se ha utilizado el dataset *weakly labeled*. Para la validación durante el entrenamiento se ha usado el dataset de *Validation* habiendo sido convertido a etiquetas débiles. Para evaluar finalmente la red neuronal se ha utilizado el dataset de *public evaluation*, en una primera instancia habiendo sido transformado a etiquetas débiles para comprobar la clasificación de etiquetas débiles y posteriormente con etiquetas fuertes.

A continuación, se muestra la distribución de las clases en los distintos dataset:

Tabla 1-Distribución clases DESED

	weakly_labeled	validation	public_evaluation
Speech	550	627	314
Dog	215	160	82
Cat	173	121	70
Alarm_bell_ringing	206	187	79
Dishes	184	171	136
Frying	171	89	88
Blender	134	80	73
Running_water	345	197	92
Vacuum_cleaner	167	91	94
Electric_shaver_toothbrush	103	62	84
Audios totales	**1573**	**1153**	**692**

La distribución de las clases dentro de cada dataset es un factor muy influyente como se verá más adelante en el análisis de resultados. Es lógico pensar que, si una clase tiene mas ejemplos que otras, la red neuronal tendrá más opciones para aprender a diferenciarla del

resto. Además, a la hora de hacer predicciones en la fase de análisis, es más fácil acertar esa clase puesto que también hay más ejemplos en el dataset de evaluación.

4 Desarrollo

En esta sección se comentarán los aspectos más importantes del desarrollo de la red neuronal. Empezando por el preprocesado de los audios para que sean interpretables por una capa de una red neuronal, pasando por el desarrollo del modelo, y acabando con el estudio posterior que se ha hecho sobre la salida de la red neuronal con el objetivo de comprobar si se ha alcanzado el objetivo principal.

4.1 Preprocesado de los audios

Como para cualquier problema de aprendizaje automático, hay que preparar el dataset para que pueda ser procesado por nuestro modelo. Es una de las decisiones más importantes que se han de tomar, puesto que una buena decisión puede suponer mejoras importantes en el resultado, al igual que una mala elección puede suponer unos pobres resultados.

Para los audios, esta decisión es menos intuitiva que para otras fuentes de datos como pueden ser las imágenes, donde están los píxeles, o el lenguaje, donde están las palabras. Para los audios existen distintas formas de representación, como puede ser la onda, véase la *Figura 7*.

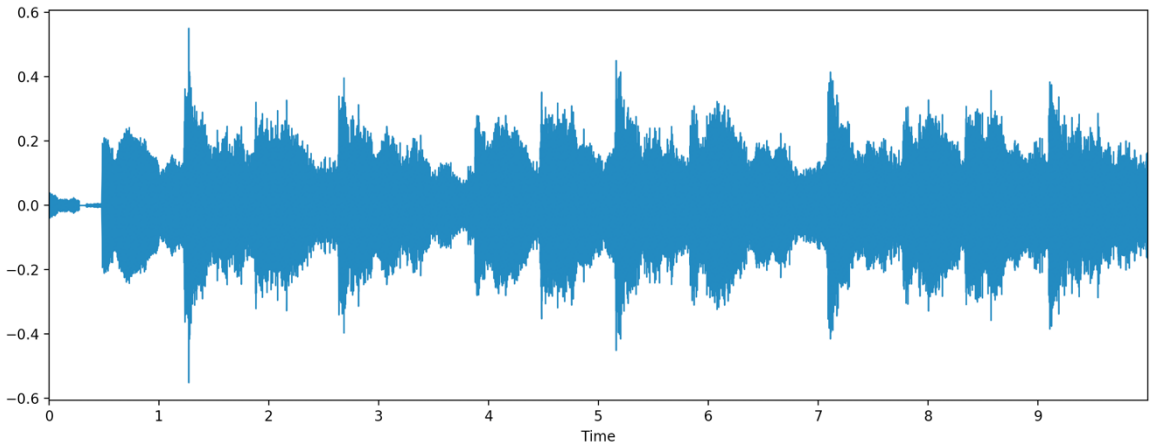

Figura 7-Onda de audio

Pero este tipo de representaciones no dan toda la información que se puede extraer de un audio. Una de las representaciones mas usadas para los audios para este tipo de propósitos son los espectrogramas [11]. Un espectrograma se centra en representar la amplitud de cada frecuencia en todos los instantes de tiempo en los que se mide. Bien se puede usar para audios como para el análisis del espectro luminoso o las ondas magnéticas [12].

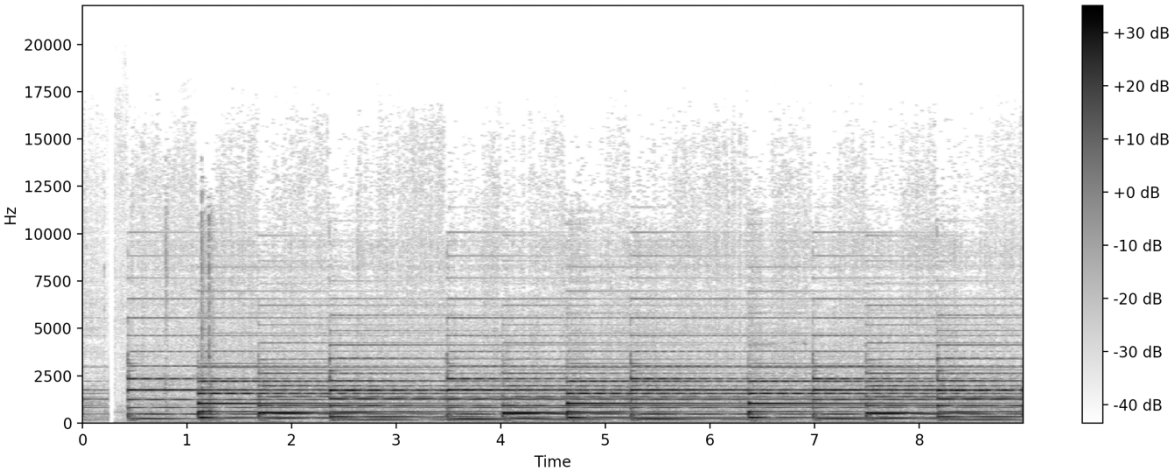

Figura 8-Espectrograma común de audio

Una de las principales corrientes de pensamiento que rodea el mundo del aprendizaje automático propone el presentar a las redes neuronales los datos de la forma más similar a como los recibe nuestro cerebro, ya que las redes neuronales se basan en el funcionamiento del mismo. Siguiendo esta línea, y teniendo en cuenta que el oído humano no percibe igual los distintos tipos de frecuencia, existen coeficientes que potencian determinadas frecuencias y crean espectrogramas que dan importancia a las frecuencias que mejor escuchan los humanos.

Se ha estudiado que los seres humanos diferencian mejor las frecuencias bajas que las altas. Un ejemplo claro para ver esto es que, dentro del espectro audible, hay cuatro octavas entre los 16Hz y los 256Hz, consideradas las octavas graves, y tres octavas entre los 2kHz y los 16kHz [13]. De esto podemos sacar la siguiente conclusión, y es que en un intervalo de 240Hz en frecuencias bajas podemos distinguir hasta 32 notas distintas, pero en un intervalo de 14kHz a frecuencias altas tan solo distinguimos 24 notas.

Es por esto que lo que se conoce como espectrograma de Mel, o melgrama, presenta el espectrograma con una escala no lineal del eje de frecuencias, dando mas importancia a la zona de bajas frecuencias, como se puede ver en las figuras 8 y 9. Ambas son espectrogramas del mismo audio, la *Figura 8* es un espectrograma común mientras que la *Figura 9* es un melgrama.

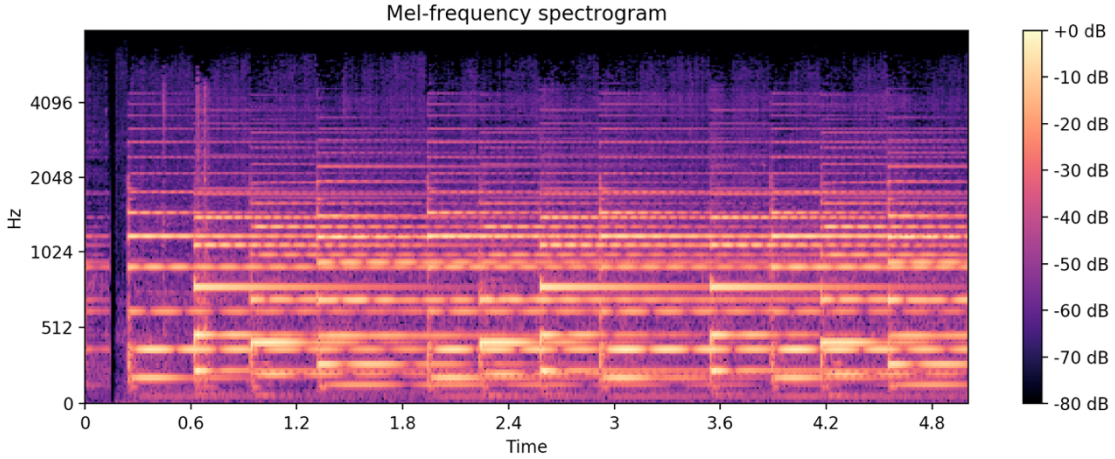

Figura 9-Espectrograma Mel

En concreto, para crear los espectrogramas del tamaño que queríamos, es decir, matrices de 64x500, asemejándonos así a los ganadores del año 2019 [14], se ha usado la librería de Python libROSA y los siguientes parámetros de la función *melspectrogram*:

- **n_mels**: número de características que queremos por cada frame, 64.
- **hop_lenght**: número de muestras de audio que queremos dejar entre los sucesivos frames, 1024. Con esto conseguimos que la longitud de los audios sea siempre inferior a 500.
- **n_fft**: longitud de la FFT, 2048.

No todos los audios tienen exactamente el mismo número de muestras, por lo que al crear los espectrogramas de Mel no acaban todos con el mismo número de frames. Dado que la entrada de nuestra red neuronal tiene que ser fija, se necesita que los espectrogramas tengan el mismo numero de frames. Redimensionar el espectrograma como si fuera una imagen no tiene sentido dado que estaríamos distorsionando el audio, de manera que ni siquiera un humano podría distinguir lo que se escucha en él. Por ello, durante la fase de transformación de audio a espectrograma, en el caso de no llegar a los 500 frames, se rellena con valores de -80dB, que es el 0 en nuestro espectrograma, como se puede ver en la *Figura 10*.

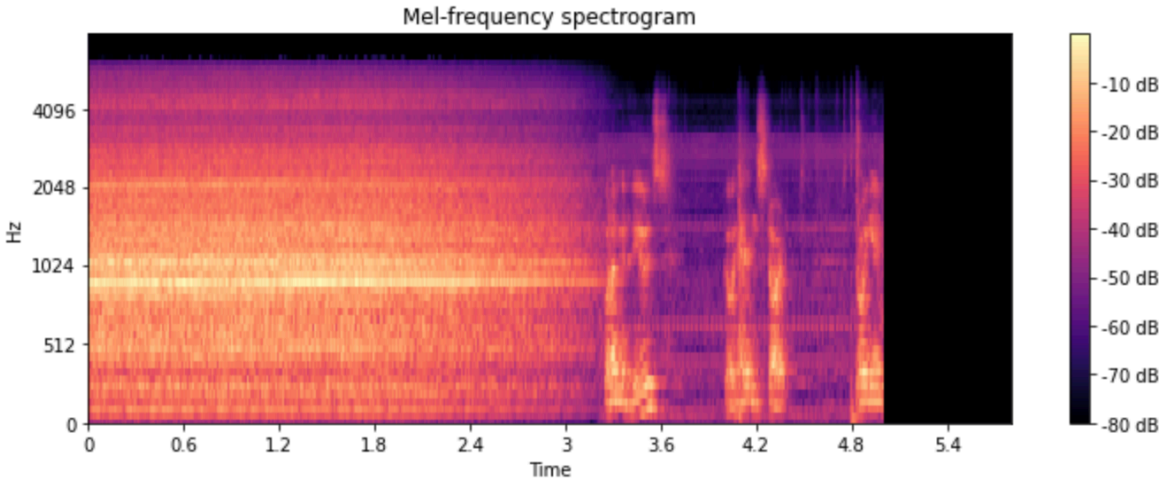

Figura 10-Espectrograma con silencio al final

4.2 Red neuronal

En esta subsección se entrará en el diseño y estructura de la red neuronal que se ha usado para llevar a cabo este trabajo. Como se adelanta en secciones anteriores, la red neuronal se trata de una *Convolutional Recurrent Neural Network*.

Este tipo de redes se caracterizan por tener unos bloques convolucionales al inicio de la red, que ayudarán a extraer las características principales de los datos. Hacia el final tienen capas recurrentes, como LSTM o GRU, que procesarán esos datos ya filtrados por las capas convolucionales e integrarán la información presente a lo largo de toda la secuencia temporal. Si nos fijamos en los modelos que se han presentado a la evaluación en años anteriores se pueden observar un gran número de proyectos que hacen uso de este tipo de redes, haciendo ver que es el estado del arte.

4.2.1 Bloque convolucional

Los bloques convolucionales se utilizan sobre todo para la clasificación de imágenes por su buen funcionamiento en ese ámbito. El principal factor por el que se usan este tipo de bloques al inicio de los modelos de clasificación de audios es porque un espectrograma, como el de la *Figura 10*, al fin y al cabo, puede ser tomado como una imagen. De hecho, se ha comprobado el buen funcionamiento de las *Convolucional Neural Networks* para la clasificación de audios, tan solo usando los espectrogramas como imágenes [15].

El bloque convolucional que se usa en este trabajo es un bloque similar al de los ganadores del año 2019 [14]. Este a su vez es un bloque muy típico, es decir, tiene los elementos más comunes de un bloque convolucional, véase la *Figura 11*. Más adelante, en esta misma sección, se entrará más en detalle en cada uno de los componentes y de los parámetros utilizados para este trabajo.

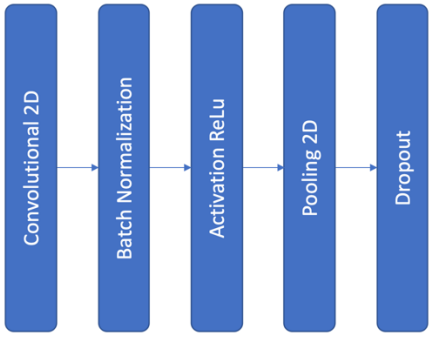

Figura 11-Bloque convolucional utilizado

Los bloques convolucionales en las CRNN se usan principalmente para hace un primer filtrado de los espectrogramas, que son datos en bruto, y así sacar las características más importantes de estos. Estas capas funcionan a base de filtros, los distintos filtros de la capa convolucional se van aplicando al espectrograma y van extrayendo así las características más relevantes. La capa convolucional es la primera capa del bloque y es la que se encarga de este proceso. Aplicando cada uno de los filtros o *kernels* se crea una nueva matriz de características a partir de la entrada, como se ve en la *Figura 12*. Estos filtros o *kernels* son en realidad los pesos de la capa convolucional y se irán reajustando durante el entrenamiento. Para la red neuronal de este trabajo se usan capas convolucionales con 64 filtros, cada uno de estos filtros con tamaño 3x3, un *padding* con valor *same*, para mantener el tamaño de las imágenes durante el filtrado, y un inicializador de pesos *glorot_uniform*. Este es un inicializador de pesos que toma muestras de una distribución uniforme dentro de *[-limit, limit]*, donde *limit = sqrt (6 / (fan_in + fan_out))* (*fan_in* es el número de unidades de entrada en el *tensor* de pesos y *fan_out* es el número de unidades de salida).

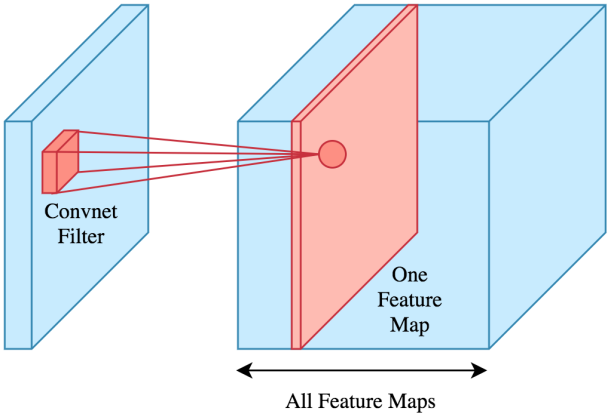

Figura 12-Filtrado convolucional

La siguiente capa dentro del bloque convolucional es una capa de normalización a nivel de *batch*. El principal motivo de añadir esta capa es que aporta efectividad al entrenamiento en general. Lo hace más rápido y también más estable, haciendo así que su rendimiento sea mucho mayor.

La tercera capa es una capa de activación, que en particular usa la función de activación ReLU. Esta función de activación se describe con la siguiente función $f(x) = \max(0, x)$ y su gráfica se muestra en la *Figura 13*. Esta función de activación se ha ido usando más en los últimos años por su buen funcionamiento y eficiencia ante el problema de desvanecimiento del gradiente. Este problema afecta a modelos que usan para entrenar el descenso de gradiente. El problema surge al añadir muchas capas a una red neuronal, algo que no se podía hacer por cuestiones de rendimiento hasta hace pocos años que ha mejorado la capacidad computacional, y consiste en que el gradiente va tomando valores cada vez más pequeños y se vuelve incapaz de modificar los valores de los pesos de la red [16].

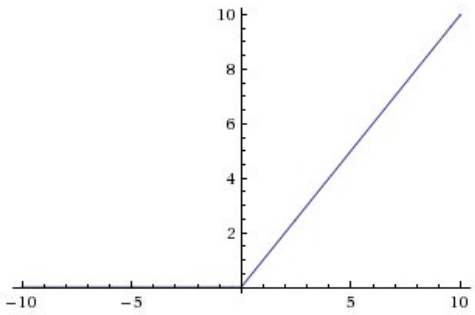

Figura 13-Grafica ReLU

La cuarta capa es también uno de los pilares de los bloques convolucionales. Se trata de la capa de *pooling*. Esta capa lo que hace es reducir la dimensionalidad de la entrada, dando así mas tolerancia a la red. Básicamente aplica ventanas de un determinado tamaño a la entrada y, dependiendo del tipo de reducción que se quiera aplicar, genera un único resultado para esa posición. Por ejemplo, en la *Figura 14* se muestra el funcionamiento de dos tipos de *pooling, Average* y *Max*. Para este trabajo se ha usado el *MaxPooling* con tamaño de

ventana 2x1, para solo ir reduciendo la dimensión en el eje de las frecuencias y no en el de los *frames*.

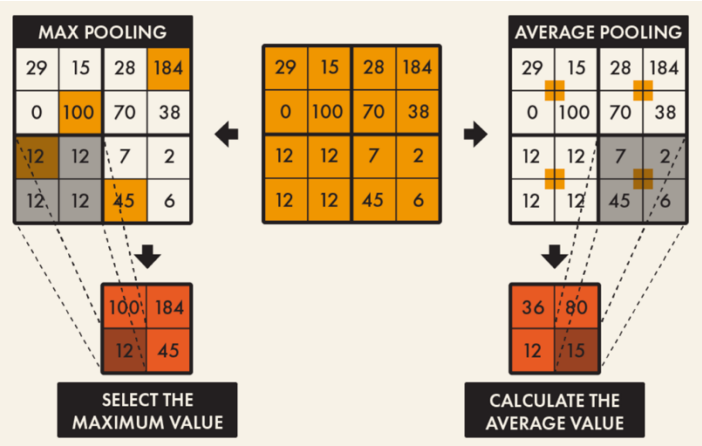

Figura 14-Funcionamiento *pooling*

Por último, la capa *Dropout* se usa para, en redes muy grandes, evitar el sobreaprendizaje. Esta capa consigue este objetivo a base de eliminar, de forma aleatoria, un porcentaje de las conexiones con la capa anterior. De esta forma se aporta a la red un poco de aleatoriedad durante el entrenamiento, forzándola a aprender mejor y a no fijarse siempre en las mismas características, proceso que puede desembocar en un sobreaprendizaje. En esta red neuronal se ha usado un porcentaje de dropout de 30%.

Este bloque se va a repetir un total de 6 veces al inicio de la red neuronal, una detrás de otra, para así conseguir ir extrayendo las principales características de todas las unidades Mel. El espectrograma de entrada tiene una dimensión de 64 unidades Mel por 500 frames. El número de filtros de las capas convolucionales siempre es 64, como las unidades Mel, pero el tamaño de la ventana de pooling es de 2x1, es decir, en cada iteración el eje se reduce a la mitad. Repite este proceso 6 veces para reducir las 64 unidades Mel a 1 solo valor, conservando la mayor parte de información en los 64 filtros de la capa convolucional. Entonces, tras las 6 iteraciones, se pasa de una dimensión 64x500 a una dimensión 1x500x64. Finalmente, después de un *reshape*, se tiene una matriz con las mismas dimensiones, 64x500, pero habiendo sido todo filtrado por la capa convolucional.

4.2.2 Bloque recurrente

Una vez se ha hecho el primer filtrado con la ayuda de las capas convolucionales, se aplica un bloque recurrente. El bloque recurrente consiste en una serie de capas especializadas en un análisis ligado al factor temporal. Esto quiere decir que es capaz de tener en cuenta la dimensión del tiempo durante el entrenamiento.

Para conseguir esto, las capas recurrentes se constituyen de lo que se llaman *cells*. Cada *cell* es una unidad de cálculo que dada una entrada genera una salida. Estas *cells* se disponen una detrás de otra para formar una cadena. A cada *cell* se le va a presentar lo que se considera un frame de nuestros ejemplos de datos. Esta cadena los irá consumiendo de principio a final e irá generando sus resultados. Además, los resultados que genere una *cell* se los pasará a la siguiente y esta los utilizará junto a su entrada normal para generar la salida que será

consumida por la siguiente *cell*. Es gracias a este método que es capaz de enlazar eventos del pasado con el futuro y generar predicciones en función de esto.

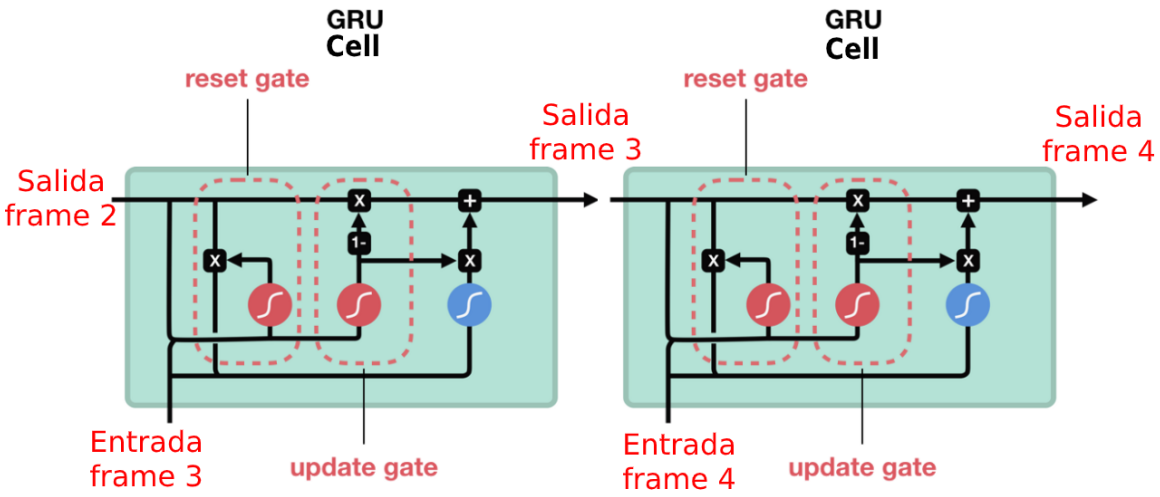

Figura 15-Funcionamiento GRU cells

Para este trabajo, de entre varias posibilidades de capas recurrentes, se ha elegido la capa GRU por ser un poco mas simple que el resto. En general la complejidad se encuentra en las *cells* de las capas. Cuantos más cálculos y entradas tiene la *cell*, más compleja es la capa y más cuesta entrenarla. En la *Figura 15* se ejemplifica el funcionamiento de la GRU y además se ve la composición de las *cells*. Como se ve en la figura, estas *cells* solo reciben como entrada el frame que les corresponde y la salida de la *cell* anterior.

Los parámetros de la capa GRU que se usan en este trabajo son los siguientes:

- **units**: 64, esto hace referencia a la dimensionalidad del espacio de salida, que se quiere mantener en 64.
- **return_sequences**: true, lo que se traduce en que a la salida nos devolverá el resultado de los 500 frames y no solo del último, como hace por defecto. Esto será necesario para más adelante aplicar el modelo de atención al audio, ya que, si solo se quedara con la salida final, no se podrían localizar los eventos de audio.
- **dropout**: 30%, igual que en los bloques convolucionales. Este dropout sirve para lo mismo que en los bloques convolucionales y es para evitar el sobreaprendizaje.

Además, a esta capa GRU se le ha añadido lo que se conoce como *wrapper*. En concreto un *bidirectional wrapper*. Básicamente lo que hace este *wrapper* es recorrer el ejemplo que se le presente en ambos sentidos, de inicio a fin y luego del final al inicio. Así no solo enlaza eventos del pasado con el futuro, sino también eventos del futuro con el pasado. De esta forma la salida del bloque recurrente tiene dimensión 128x500. El 128 viene de haberse duplicado, por ser bidireccional, el 64 y se mantienen los 500 frames al tener *return_sequences* a *true*.

4.2.3 Bloque de atención

El mecanismo de atención utilizado en este trabajo se compone de una capa densa seguida de una capa *softmax*. Una vez obtenemos la salida de la capa GRU bidireccional, se procesará a través de una capa densa de 10 neuronas. La capa densa tiene este número de neuronas porque es el número de clases objetivo que tiene el problema y por tanto es el número de vectores de atención que se quieren estudiar más adelante. De esta forma, la salida de la capa

densa será una matriz de 10x500, es decir, se tendrán 10 vectores, uno por clase, de 500 frames cada uno. Después se aplica la capa *softmax* para normalizar los vectores.

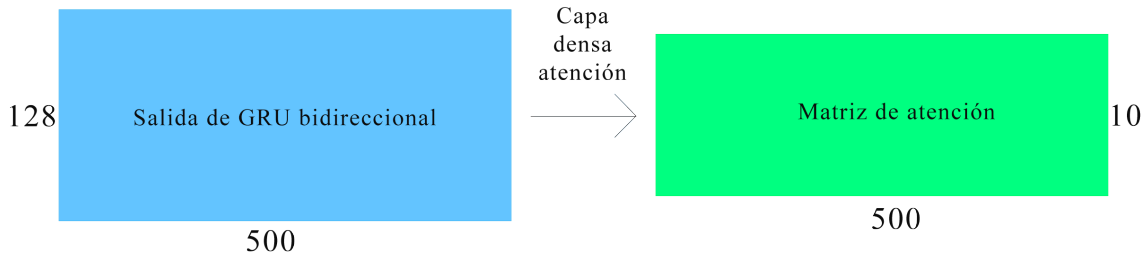

Figura 16-Salida de GRU a matriz de atención

Con el objetivo de que cada uno de los vectores se especialice en una clase en concreto, clonamos 10 veces la salida de la GRU bidireccional y las concatenamos con cada uno de los vectores. De esta forma se utilizará cada una de estas nuevas matrices para predecir una clase. Cada una de estas matrices atravesará una capa densa de una neurona que generará un vector de 1x500 y se condensará en un único valor a través de un *global average pooling*. Este valor será la predicción de la red para esa clase en concreto.

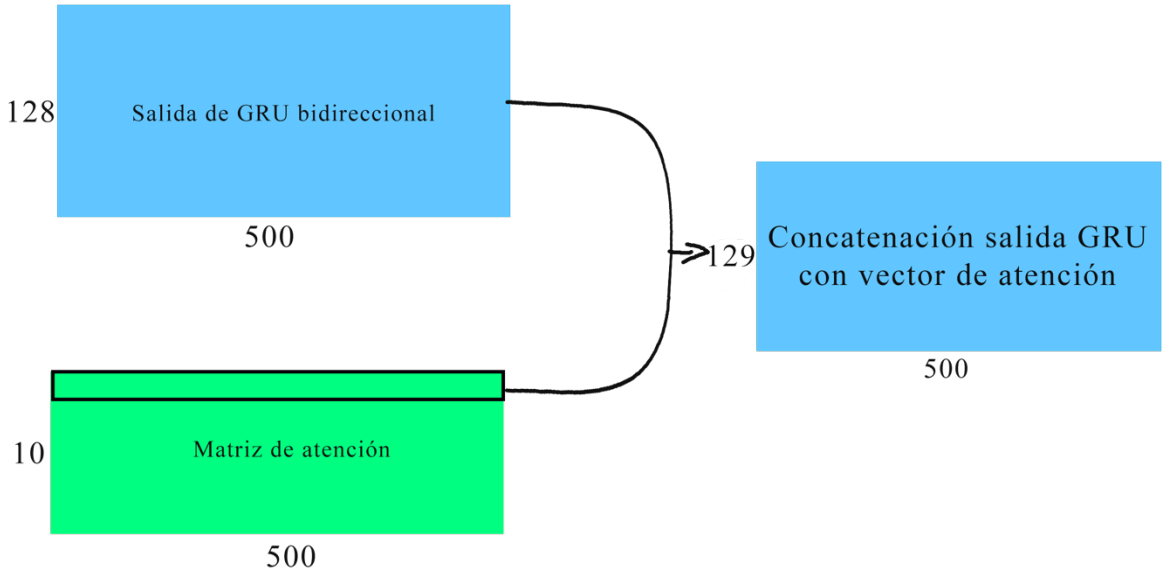

Figura 17-Concatenación vectores de atención con GRU

4.3 Entrenamiento

Al tratarse de un problema en el que se puede encontrar más de una clase en un único ejemplo, se tendrá que binarizar las clases. A diferencia de un problema en el que un ejemplo solo puede corresponder a una única clase, en este problema en un solo audio puede haber cero, una, dos o más ocurrencias de clases. Es por esto que el objetivo de la red será predecir un vector de 10 elementos. Estos elementos serán 0 o 1 dependiendo de si la clase correspondiente se da o no en el ejemplo. En consecuencia, la función de coste que se tendrá que utilizar a la hora de compilar el modelo será *binary cross-entropy*. De esta forma, la red

neuronal contará con la función de coste adecuada que evaluará cómo de correctas son las clasificaciones binarias que hace. Esta función de coste subdivide el problema de clasificación en tantos subproblemas de clasificación binaria como clases haya. De esta forma la clasificación a 0 o 1 de una clase es completamente independiente de el resto de clases. Esto es justamente lo que se quiere, ya que no se presupone ningún tipo de relación de ocurrencia entre las clases.

Además, se usa una reducción del coeficiente de aprendizaje para evitar el sobreaprendizaje. Este algoritmo se fija en el *validation_loss* para decidir cuando reducir el coeficiente de aprendizaje. Si se suceden varias épocas en las que el *validation_loss* apenas varía, el coeficiente de aprendizaje se reduce. Esto es así porque presupone que se ha llegado a un punto máximo del aprendizaje y quiere conservar los ajustes de los pesos.

El número de épocas del entrenamiento es objeto de estudio en el análisis de los resultados, ya que es un punto importante a la hora de obtener los resultados esperados.

5 Análisis de los resultados

En esta sección se analizarán los resultados obtenidos durante el trabajo. Se compararán distintas redes neuronales y se explicarán las diferencias entre ellas y el porqué de sus resultados.

5.1 Análisis resultados etiquetas débiles

En esta subsección se comentarán los resultados obtenidos en la tarea de predicción de etiquetas débiles. Como se comenta en secciones anteriores, el objetivo de este trabajo es ver cómo a través de un entrenamiento con etiquetas débiles y el uso de modelos de atención se puede hacer un estudio a posteriori de determinadas capas que componen la red neuronal y hacer una aproximación al problema de localización de eventos dentro de los audios. Para ello, a continuación, se mostrarán los resultados de los distintos modelos comparados. Pero antes, introduciremos los modelos a comparar.

- Modelo 2019: el modelo 2019 se corresponde con el modelo utilizado por los ganadores de la evaluación de la tarea 4 de DCASE en el año 2019. Era la evaluación inmediatamente anterior a la de este trabajo y este era el modelo con mejor resultado, por lo que es un buen punto de partida y de comparación.
- Modelo sin atención: modelo creado para este trabajo a partir del modelo de 2019 incluyendo algunos cambios. Este modelo es idéntico al que más tarde se le añadirá el bloque de atención. El mayor cambio introducido es en la parte convolucional, donde, como se explica en la sección 4.2.1, este modelo tiene 6 bloques. Esto es así porque se va reduciendo la dimensión a la mitad en cada bloque hasta llegar a 1 y partimos de una dimensión de 64. En el modelo original de 2019 se hacía en 3 bloques ya que se reducía a un cuarto la dimensión en cada bloque.
- Modelo con atención: este modelo es igual que el anterior excepto porque incluye el modelo de atención además de los cambios oportunos a la salida de la red para adaptar las salidas de las últimas capas a los vectores objetivo como se explica en la sección 4.2.3.

Tabla 2-Comparativa de resultados de los distintos modelos

Accuracy set entrenamiento

	1	2	3	4	5	Media	Desv. Estandar
Modelo 2019 100 epocas	0,7482	0,5177	0,7224	0,4410	0,6476	0,61538	**0,132365335**
Modelo 2019 150 epocas	0,6622	0,7737	0,6831	0,7395	0,5988	0,69146	0,068160054
Sin atencion 100 epocas	0,9036	0,8593	0,9397	0,8878	0,7915	0,87638	0,055626226
Sin atencion 150 epocas	0,9550	0,7598	0,8441	0,7718	0,9645	0,85904	0,097484271
Con atencion 100 epocas	0,8853	0,8998	0,8948	0,8821	0,9202	0,89644	0,015068610
Con atencion 150 epocas	0,9626	0,8017	**0,9683**	0,8897	0,9670	**0,91786**	0,072881294

Accuracy set validacion

	1	2	3	4	5	Media	Desv. Estandar
Modelo 2019 100 epocas	0,4431	0,3313	0,4405	0,3026	0,3998	0,38346	**0,063907144**
Modelo 2019 150 epocas	0,3842	0,4527	0,4275	0,3954	0,3980	0,41156	0,028018619
Sin atencion 100 epocas	0,4752	0,4535	**0,5091**	0,4787	0,4171	**0,46672**	0,034089617
Sin atencion 150 epocas	0,4744	0,3833	0,4128	0,4067	0,5013	0,43570	0,049780066
Con atencion 100 epocas	0,4336	0,4475	0,4457	0,4509	0,4986	0,45526	0,025090496
Con atencion 150 epocas	0,4640	0,4105	0,4718	0,4232	0,4874	0,45138	0,032935422

Accuracy set evaluacion

	1	2	3	4	5	Media	Desv. Estandar
Modelo 2019 100 epocas	0,4147	0,3294	0,4364	0,3063	0,3540	0,36816	**0,055570163**
Modelo 2019 150 epocas	0,3526	0,4609	0,3757	0,4234	0,3526	0,39304	0,047692169
Sin atencion 100 epocas	0,4638	0,4089	0,5014	0,4523	0,4248	0,45024	0,035922319
Sin atencion 150 epocas	0,4768	0,4147	0,3713	0,3973	0,4942	0,43086	0,052576734
Con atencion 100 epocas	0,4017	0,4826	0,4320	0,4624	0,4768	0,45110	0,033858529
Con atencion 150 epocas	0,4494	0,4723	**0,4884**	0,4450	0,4638	**0,46378**	0,017594090

mAP set evaluacion

	1	2	3	4	5	Media	Desv. Estandar
Modelo 2019 100 epocas	0,4663	0,3982	0,4592	0,3401	0,4180	0,41636	**0,051179420**
Modelo 2019 150 epocas	0,4224	0,5171	0,4450	0,4679	0,4429	0,45906	0,036226965
Sin atencion 100 epocas	0,5107	0,4601	**0,5339**	0,4877	0,4847	0,49542	0,028000929
Sin atencion 150 epocas	0,5207	0,4632	0,4634	0,4320	0,5296	0,48178	0,041721122
Con atencion 100 epocas	0,4692	0,5097	0,4874	0,5048	0,5147	0,49716	0,018710238
Con atencion 150 epocas	0,4936	0,4902	0,5186	0,5004	0,4974	**0,50004**	0,011066526

En la tabla inmediatamente superior se ven cuatro tablas. Cada una hace referencia a un tipo de métrica en un determinado set. Para cada uno de los modelos y número de épocas se han hecho 5 experimentos de los que luego se ha sacado la media y la desviación estándar. El *accuracy* es el porcentaje de acierto absoluto de las predicciones, es decir, la igualdad total entre la etiqueta objetivo y la predicha por la red. Al ser un problema en el que puede haber más de una clase en cada ejemplo se da el caso de que en un audio con dos clases, la red acierte una y no la otra, contando esta situación como una predicción fallida. Es por esto que, en el set más relevante, el de evaluación, se incluye también la medida *mean Average Precision* que tiene en cuenta el tipo de casuística explicada anteriormente y ofrece un porcentaje de acierto más real y menos absoluto.

En primer lugar, se ve la mejora clara que supone el dividir la parte convolucional en más bloques. Se ve una gran mejoría entre el modelo de 2019 y el modificado para este trabajo en todas las tablas. El hacerlo pasar por mas bloques convolucionales hace que la extracción de características sea más pausada y precisa, al no perder tanta información en cada bloque. La peor precisión en la predicción de clases también va acompañada de una desviación más grande, lo que permite pensar que el modelo no es muy fiable en cuanto a que en cada experimento da métricas más dispares y tiene menos continuidad.

En general el número de épocas mejora el resultado de las métricas menos en el modelo sin atención. Esto hace pensar que el límite de este modelo está en un número de épocas más bajo y que el modelo de atención lo que hace es ampliar este umbral, permitiendo a la red evolucionar e ir mas allá con un número superior de épocas y alcanzando en general mejores resultados, aunque sean de un 1%.

A fin de cuentas, observando las métricas del set de evaluación en la comparativa, se puede determinar que el mejor modelo es el que incluye bloque de atención y con 150 épocas. Este será el modelo que se usará en la siguiente sección, donde se estudiará la salida del mecanismo de atención para ver si se puede extraer información para localizar eventos y su precisión.

5.2 Análisis resultados etiquetas fuertes

Como se introduce en otras secciones, el propósito de este trabajo es el de estudiar la posibilidad de entrenar una red neuronal con etiquetas débiles y ser capaces de alcanzar una aproximación a la localización de eventos de audio. Este objetivo se pretende alcanzar con la ayuda de los modelos de atención. El estudio y análisis que se hace en esta sección corresponde al análisis del modelo de atención entrenado en 150 épocas.

Para poder obtener las salidas del modelo de atención tal y como se quiere, se crea un nuevo modelo a partir de las capas ya entrenadas. Lo que se hace es cambiar la capa de salida y hacer que sea la capa softmax del bloque de atención. El bloque de atención se componía de una capa densa que es la que genera la matriz de atención y luego de una capa softmax que se utiliza para normalizar los valores. A la salida de la capa softmax le damos el nombre de matriz de atención, ya que se compone de los 10 vectores de atención correspondientes a cada clase. Esta será la salida que se analizará de aquí en adelante.

Los audios que se utilizan para estas pruebas son los de set de evaluación, es decir, la red no se ha entrenado con estos audios.

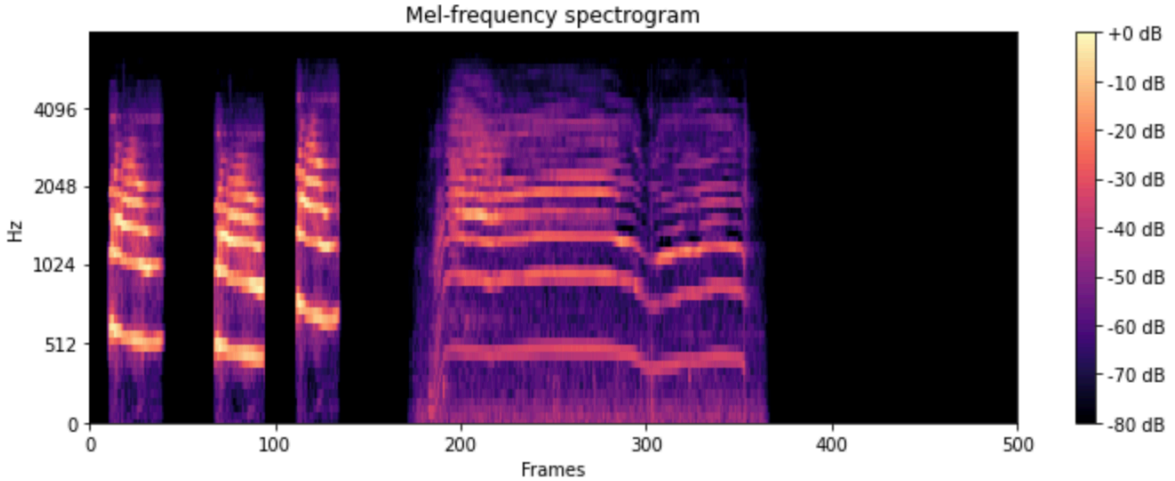

Figura 18-Espectrograma audio de *Cat*

La *Figura 18* es el espectrograma de un audio del set de evaluación etiquetado como *Cat*. Se ha elegido este audio como primer ejemplo puesto que es una grabación muy limpia donde es fácil identificar por el ojo humano donde se posicionan los eventos de audio del

maullido. Como apunte, en concreto este audio ha sido correctamente clasificado por la red neuronal.

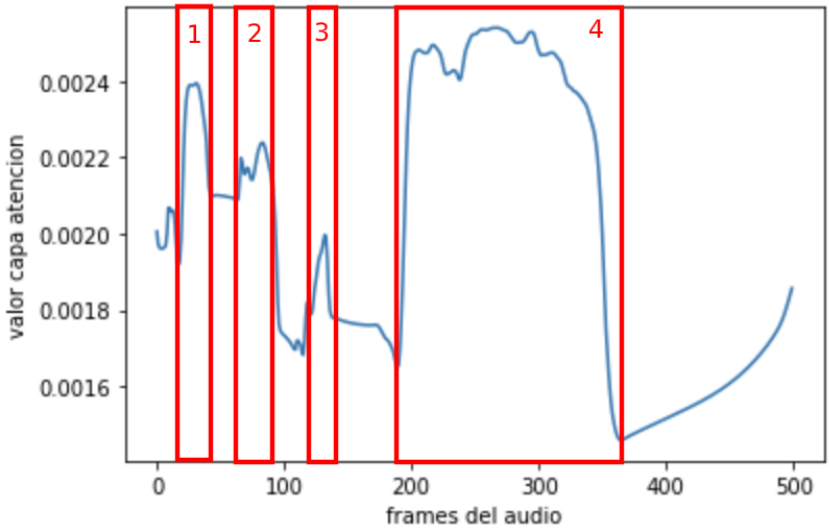

Figura 19-Gráfica vector de atención clase *Cat* del audio *Cat*

Si se observa la matriz de salida de la capa de atención y en concreto en el vector de atención que corresponde con la clase *Cat* se ve que se dibujan los momentos en los que ocurren los maullidos como se ve en la *Figura 19*. El problema viene cuando se observa la matriz de atención en conjunto.

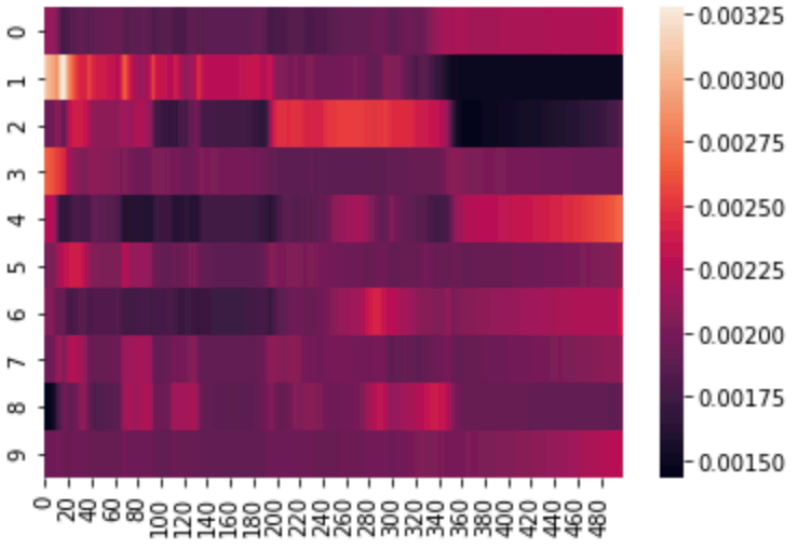

Figura 20-Matriz de atención de audio *Cat*

Al observar la matriz de atención, es decir, la salida de la capa softmax del bloque de atención, en conjunto, se ve que no solo el vector 2, que es el que se corresponde con la clase *Cat* se activa o dibuja las ocurrencias de los eventos. Si bien es el vector 2 el que dibuja más claramente el ultimo maullido, los vectores 5, 7 y 8 también dibujan los maullidos del principio de forma bastante clara. Esto hace pensar que de alguna manera los vectores no se

están especializando del todo en su clase, o que de alguna forma aprenden a fijarse en el resto de clases para determinar que no son la suya y así no dar falsos positivos.

De alguna forma esto no supone un problema para la localización de eventos de audio en ejemplos donde tan solo se dé una clase, dado que todas las activaciones se deberían corresponder con la única clase que hay dentro de la grabación, como es el caso del audio *Cat* que se acaba de presentar. El problema se agranda cuando tenemos mas de una clase dentro de un audio.

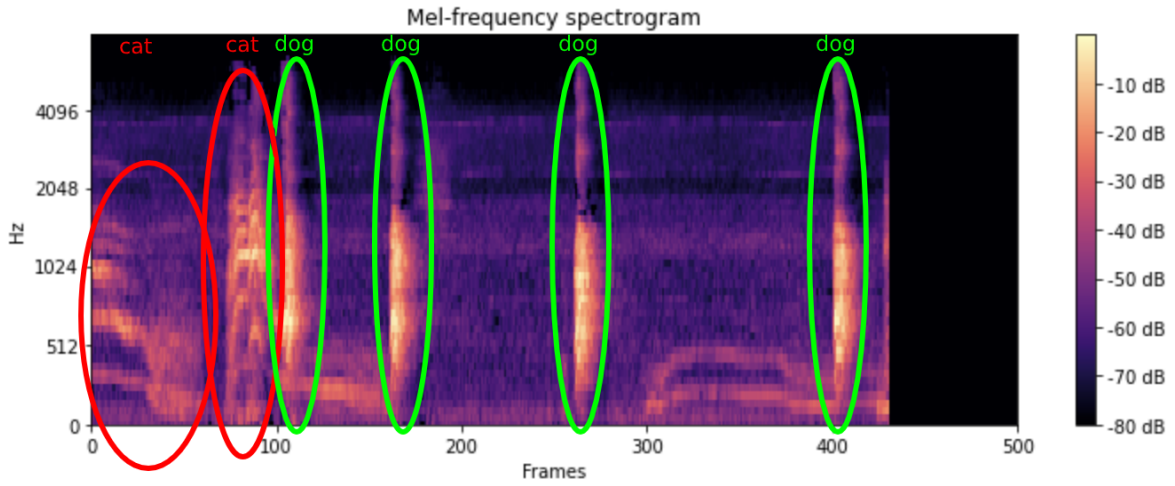

Figura 21-Espectrograma de audio *Cat* y *Dog*

En el espectrograma de la *Figura 21* se dan dos clases, *Cat* y *Dog*. De nuevo se escoge este ejemplo porque es fácil ver donde empiezan y acaban cada una de ellas. Este audio también ha sido bien clasificado por la red neuronal, por lo que se presupone que tiene un buen entendimiento de lo que ocurre dentro del audio.

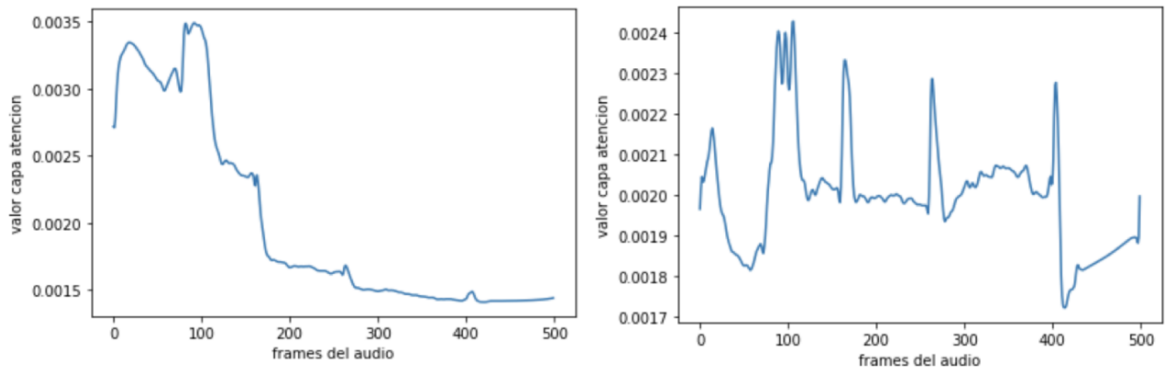

Figura 22-Gráfica vector atención audio *Cat(izq)* y *Dog(der)*

Como se ve en la gráfica del vector de atención de *Cat*, *Figura 22 izquierda*, aunque predomina el momento donde se escuchan los maullidos, los ladridos del perro también se dibujan en la gráfica y la distorsionan. Por el otro lado, si observamos la gráfica del vector de atención de *Dog*, *Figura 22 derecha*, se ve como el primer maullido se dibuja claramente en la gráfica al principio, aunque luego los ladridos se dibujan con más fuerza.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

A modo de conclusión se puede afirmar que se ha conseguido el objetivo de crear una red neuronal que, con la imprescindible ayuda de un mecanismo de atención, es capaz de abstraerse y localizar eventos de audio a pesar de no haber sido entrenada para ello. A modo de resumen se alcanzan las siguientes conclusiones y objetivos cumplidos a lo largo de todo el trabajo:

- Se ha adaptado una red neuronal preexistente al trabajo para que funcionase según se requería, incluyendo en su estructura un mecanismo de atención. De esta forma se ha podido comprobar de mejor manera de que forma impactarían los mecanismos de atención en los modelos típicos de clasificación de audios.

- En la adaptación de la red neuronal se cambió la parte convolucional, haciendo que esta tuviese más bloques y que la reducción de dimensionalidad que se hacia con las capas de *pooling* fuera más progresiva. Esto, como se ha visto reflejado en los resultados, supone una gran mejora.

- A pesar de no ser el objetivo de este trabajo, en el análisis de resultados se ha podido comprobar una ligera mejora en el desempeño de los modelos al incluir dentro de ellos el mecanismo de atención. Esto sirve de antesala del posible buen resultado en el objetivo real del trabajo, que es localizar los eventos de audio dentro de la grabación.

- Como objetivo mas importante de este trabajo se tiene la localización de eventos de audio dentro de las grabaciones con ayuda de los mecanismos de atención. Como se puede ver en la sección de análisis de resultados, este objetivo se logra a grandes rasgos. En la sección mencionada se puede ver como en la salida del mecanismo de atención se enfatizan los momentos en los que se dan los eventos de audio. Esto ayuda tanto a la red a clasificar los eventos, como al objetivo de localizar dentro de la grabación las marcas temporales donde ocurren los eventos.

- Dentro del marco de que el objetivo principal se ha cumplido, hay que recalcar los problemas encontrados. Principalmente, y el que más afecta a la localización de eventos, es el tema de que los vectores de atención generados no son capaces de enfocarse tan solo en su clase, más bien se fijan en todas las clases. La conclusión a la que se llega es que el mecanismo de atención quiere que los vectores se fijen en todas las clases para que, a la hora de clasificar una nueva grabación, se determine con la misma seguridad la clasificación positiva como negativa para esa clase. Por ejemplo, el vector que corresponde con la clase *Cat* enfatiza también, aunque menos, los eventos *Dog* para poder determinar que esos eventos no se corresponden con su clase.

6.2 Trabajo futuro

Como trabajo para el futuro, el modelo presentado en este trabajo cuenta con el problema de no ser capaz de hacer que cada vector de atención se centre al cien por cien en su clase. Este problema puede agravarse al incluir más clases al problema o más clases en un solo ejemplo. Por ello resultaría interesante continuar este trabajo con el objetivo de centrar cada vector en su clase, alcanzando un mejor resultado.

También se podría experimentar con otros mecanismos de atención mas complejos creados para otros campos. El mecanismo de atención utilizado en este trabajo es relativamente sencillo teniendo en cuenta lo que se puede llegar a crear. Hay mecanismos de atención que son modelos en si mismos.

Además de estas propuestas concretas, que solo tiene relación con este trabajo, se anima a continuar investigando las aplicaciones de los mecanismos de atención en este y todos los campos de la inteligencia artificial. Tienen un gran potencial aún por descubrir y ya han supuesto un hito en otros campos.

Bibliografía

[1] Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In Competition and cooperation in neural nets (pp. 267-285). Springer, Berlin, Heidelberg.

[2] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[3] Torres, J. (2018). DEEP LEARNING Introducción práctica con Keras. Lulu. com.

[4] D. Calvo, «Red Neuronal Recurrente,» 9 12 2018. [En línea]. Available: http://www.diegocalvo.es/red-neuronal-recurrente/. [Último acceso: 31 5 2020].

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[6] Dcase Community, «DCASE,» [En línea]. Available: http://dcase.community/. [Último acceso: 20 4 2020].

[7] Piczak, K. J. (2015, October). ESC: Dataset for environmental sound classification. In Proceedings of the 23rd ACM international conference on Multimedia (pp. 1015-1018).

[8] Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017, March). Convolutional recurrent neural networks for music classification. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2392-2396). IEEE.

[9] Turpault, N., Serizel, R., Salamon, J., & Shah, A. P. (2019). Sound event detection in domestic environments with weakly labeled data and soundscape synthesis.

[10] Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., ... & Ritter, M. (2017, March). Audio set: An ontology and human-labeled dataset for audio events. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 776-780). IEEE.

[11] Wyse, L. (2017). Audio spectrogram representations for processing with convolutional neural networks. arXiv preprint arXiv:1706.09559.

[12] Wikipedia, «Espectro de frecuencias,» [En línea]. Available: https://es.wikipedia.org/wiki/Espectro_de_frecuencias. [Último acceso: 24 4 2020].

[13] Wikipedia, «Espectro audible,» [En línea]. Available: https://es.wikipedia.org/wiki/Espectro_audible. [Último acceso: 24 4 2020].

[14] Lin, L., Wang, X., Liu, H., & Qian, Y. (2019). Guided learning convolution system for dcase 2019 task 4. arXiv preprint arXiv:1909.06178.

[15] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., ... & Slaney, M. (2017, March). CNN architectures for large-scale audio classification. In 2017 ieee international conference on acoustics, speech and signal processing (icassp) (pp. 131-135). IEEE.

[16] Wikipedia, «Problema de desvanecimiento de gradiente,» [En línea]. Available: https://es.wikipedia.org/wiki/Problema_de_desvanecimiento_de_gradiente. [Último acceso: 27 4 2020].

Glosario

CRNN	Convolutional Recurrent Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neuronal Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
DCASE	Detection and Classification of Acoustic Scenes and Events
Kernel	Matriz de pesos con los que una CNN extrae características
Cell	Modelo matemático que conforma la unidad básica de una capa recurrente
Python	Lenguaje de programacion Python
libROSA	Libreria de Python para audio y musica