UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Modelos de aprendizaje profundo con auto-atención para detección de eventos de audio

Julio González Herrero
Tutor: Doroteo Torre Toledano

MAYO 2021

Modelos de aprendizaje profundo con auto-atención para detección de eventos de audio

AUTOR: Julio González Herrero
TUTOR: Doroteo Torre Toledano

Grupo AUDIAS – Audio, Data Intelligence and Speech
Dpto. Tecnología Electrónica y Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2021

Resumen (castellano)

Este Trabajo Fin de Grado tiene como objetivo la implementación de redes neuronales basadas en mecanismos de autoatención para la detección y clasificación de eventos de audio.

El sistema desarrollado está inspirado en la implementación que obtuvo los mejores resultados en la evaluación competitiva DCASE2020, específicamente la cuarta tarea, dedicada a la clasificación de datos débilmente etiquetados. Estas evaluaciones tienen como objetivo incentivar el desarrollo o implementación de técnicas que mejoren el campo de investigación al que corresponden.

Se ha utilizado el conjunto de datos DESED (Domestic Environment Sound Event Detection), formado por sonidos grabados y sintéticos. Estos archivos de audio contienen sonidos presentes en un entorno doméstico, tales como un cepillo de dientes, perros, gatos o habla humana. Los sonidos están presentes como sonidos principales del audio y como sonidos de fondo. Lo datos sintéticos están formados por combinaciones aleatorias de sonidos principales y de fondo.

La implementación utiliza la técnica de aprendizaje Mean Teacher. En este método se entrena un modelo profesor en paralelo al modelo principal o estudiante. Las predicciones realizadas por el modelo profesor se utilizan como etiquetas para los datos no etiquetados al calcular la pérdida del modelo estudiante.

Las redes implementadas son la red Transformer, que utiliza un módulo de autoatención para relacionar elementos individuales de una secuencia con el resto de esta, y la red Conformer, que combina el mismo mecanismo de autoatención con redes convolucionales para relacionar elementos con otros cercanos en la secuencia. La red Conformer fue la que obtuvo mejores resultados en la cuarta tarea de la evaluación DCASE2020.

Tanto la red Transformer como la red Conformer son redes complejas que se utilizan en problemas de transformación de secuencias en secuencias (en nuestro caso para transformación de secuencias de audio en secuencias de etiquetas). Para la red Transformer existen implementaciones libres en repositorios como GitHub, pero no ocurre lo mismo para la red Conformer en el momento de desarrollar el trabajo, debido a su novedad, por lo que una de las mayores dificultades del trabajo ha consistido en implementar estas redes.

Para la implementación de estas redes se han implementado dos módulos, que imitan la estructura ideada para la evaluación DCASE2020. Esta estructura implementa una red convolucional como módulo de extracción de características, y un clasificador por posición.

Adicionalmente se añade un elemento extra al principio de cada secuencia de audio, que se utilizará para ayudar en la clasificación.

Abstract (english)

The objective of this Bachelor Thesis is to implement neural networks based on self-attention mechanisms for sound event detection and classification.

The system that has been developed is based on the implementation designed for the DCASE2020 challenge, specifically the fourth task, which focuses on classification of weakly labeled data. These challenges are intended to incentivize research and implementation of techniques to improve their corresponding research field.

This implementation utilizes the Mean Teacher learning technique. This method trains a teacher model alongside the main model, also called student model. The predictions made by the teacher model are used as labels when calculating the student models loss.

The implemented networks are the Transformer network, which includes a self-attention module to relate individual elements of a sequence with the rest of it, and the Conformer network, which combines the same self-attention mechanism with convolutional networks to relate elements of the sequence with other nearby ones. The conformed network obtained the best results in the fourth task of the DCASE2020 evaluation.

Both the Transformer net and the Conformer net are complex networks utilized in problems relating to sequence-to-sequence transformations (in our case to transform audio sequences into label sequences). There are free implementations of the Transformer net available in in free repositories such as Github, but the same is not true for the Conformer net as of the development of this thesis, due to its novelty, thus one of the greatest difficulties of this thesis has been the implementation of this networks.

To implement these networks two modules have been implemented, imitating the structure ideated for the DCASE 2020 evaluation. This structure implements a convolutional network as a feature extraction module, and a position-wise classifier.

Additionally, an extra element is added to the beginning of each audio sequence, to be used in classification.

Palabras clave (castellano)

Redes neuronales, atención, transformer, conformer, audio, detección de eventos de audio.

Keywords (english)

Neural networks, attention, transformer, conformer, audio, sound event dettection.

Agradecimientos

Quiero dar gracias a mi familia por su apoyo a lo largo de este periodo.

También me gustaría dar las gracias a los miembros del grupo AUDIAS por su ayuda con este trabajo. En especial a Doroteo Torre por guiarme a lo largo del proceso y darme consejos cuando los he necesitado.

Muchas gracias.

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

Los mecanismos de atención son un desarrollo relativamente reciente que ha generado muy buenos resultados en diferentes campos del aprendizaje automático. En su uso en modelos de traducción secuencia a secuencia, la atención calcula la relación entre las partes de dos secuencias de datos, permitiendo crear vectores de contexto que ponderan con mayor peso las partes más relevantes de la secuencia para obtener la palabra a traducir en cada momento.

La atención ha supuesto avances importantes en campos tales como la clasificación de imágenes, la traducción de texto, y el procesamiento de audio.

En cuanto al campo de la clasificación de audio, debido a varias limitaciones a la hora de obtener datos, los avances realizados han sido lentos. Es por esto que avances como la atención, que permiten extraer más información de datos poco claros, son importantes.

1.2 Objetivos

El objetivo de este TFG es intentar replicar los resultados obtenidos por el mejor sistema en la tarea cuatro del desafío DCASE2020 [12], que se basaba en redes conformer, una evolución de las redes transformer que incluye capas convolucionales y mediante estos resultados analizar la efectividad de los mecanismos de atención en el aprendizaje automático para la clasificación de eventos de audio.

Para esto se han programado dos redes neuronales, un modelo Transformer y un modelo Conformer, y se ha modificado el programa baseline de la tarea para insertarlos.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Introducción:** El presente capítulo.
- **Estado del arte:** Describe el campo de la detección de eventos de audio, incluyendo los conjuntos de datos más utilizados y los modelos que han dado mejores resultados a lo largo de su historia.
- **Diseño:** Contiene una descripción del modelo sobre el que se ha implementado el proyecto, el conjunto de datos utilizado y otros detalles de la implementación, así como de los criterios que se han utilizado para evaluar los resultados.
- **Desarrollo:** En este capítulo se describen en más detalle los modelos implementados, así como la estructura de la red común a ambos, y las modificaciones realizadas sobre el modelo base para poder implementar los modelos.
- **Integración, pruebas y resultados:** Contiene los resultados obtenidos de las pruebas realizadas sobre los modelos, así como una comparación entre ellas y el modelo base.
- **Conclusiones:** Este apartado contiene las conclusiones del trabajo.

2 Estado del arte

2.1 Detección de eventos de audio

Se puede definir la detección de eventos de audio como el proceso de identificar las fuentes de sonido en una grabación de audio, junto con su punto de inicio y fin en el tiempo. Tiene gran utilidad a la hora de analizar lo que ocurre en un entorno a partir de los sonidos que se producen en él, por ejemplo, al analizar grabaciones de seguridad, el estudio de ecosistemas, o la anotación automática de conjuntos de datos.

Es muy común que una grabación de audio contenga múltiples eventos que coinciden en el tiempo. La detección de eventos en este tipo de grabaciones se conoce como detección polifónica.

Este campo ha tenido dificultades a la hora de avanzar debido a la falta de conjuntos de datos de tamaño considerable que permitan experimentar con diferentes técnicas. La creación, más reciente, de nuevos conjuntos de datos ha permitido nuevos desarrollos.

2.1.1 Evaluaciones competitivas

Las evaluaciones competitivas son competiciones en las que se presenta un problema para el cual los participantes deben presentar una solución. El ganador es aquel que presente la solución que obtenga los mejores resultados. El objetivo de estas competiciones es motivar la investigación y aplicación de nuevas tecnologías en un campo concreto, y permitir comparar los diferentes avances tecnológicos utilizados por los participantes.

CLEAR [11] fue un estudio pionero realizado en 2006 por varias universidades centrado en sonidos producidos por actividades humanas, tales como voces, aplausos, o sonidos de objetos cotidianos como puertas. Junto con la clasificación de eventos acústicos, trabajaron también en la detección de eventos acústicos que exige, además de reconocer determinados sonidos, localizarlos en el tiempo. En la evaluación el equipo creó varias bases de datos con varios cientos de ejemplos, concretamente dos cuyos ejemplos eran sonidos aislados, y una formada a partir de grabaciones de seminarios. Los ejemplos podían pertenecer a 14 clases distintas, aunque sólo se evaluaron 12. Para modelar los datos se usaron dos sistemas basados en modelos ocultos de Markov (HMMs) y otro basado en máquinas de vectores soporte (SVMs), modelos que no se tratarán en este TFG.

DCASE (Detection and Classification of Acoustic Scenes and Events) es una comunidad que organiza desafíos anuales relacionados con la clasificación de audio. Estos desafíos se centran en el reconocimiento de sonidos en entornos reales, donde las fuentes de audio se solapan, y el entorno distorsiona los sonidos. Cada año se propone un número de tareas, cada una centrada en un aspecto diferente del campo. Por ejemplo, algunas tareas se centran en el reconocimiento de múltiples tipos de sonido en un solo archivo de audio, mientras que otras están enfocadas al reconocimiento de la fuente de audio en el espacio.

La cuarta tarea del desafío DCASE2020 se centra en la clasificación de grandes números de eventos de audio débilmente etiquetados por clase y por segmento.

2.1.2 Modelos utilizados por el baseline DCASE2020

Antes de la publicación del artículo *Attention is all you need* [9], los mejores resultados a la hora de clasificar eventos de audio se obtenían mediante la combinación de modelos convolucionales y modelos recurrentes como las LSTM (Long Short-Term Memory), con modelos codificador-decodificador.

2.1.2.1 Redes Neuronales Convolucionales (CNN)

Las redes convolucionales son redes neuronales que incluyen capas especiales conocidas como capas de convolución y capas de agrupación (*pooling*). Las capas de convolución realizan la operación de convolución matemática sobre las matrices de datos, extrayendo las características de diferentes regiones de la matriz en forma de mapas de características. Estos mapas son introducidos a las capas de agrupación, que filtra las características presentes para reducirlas a las más significativas. El resultado de estas operaciones es habitualmente procesado por una red de capas completamente conectadas, que extrae los resultados finales.

2.1.2.2 Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes son redes en las que el resultado de procesar un elemento de la secuencia se reutiliza al analizar el siguiente elemento. Esto se consigue reciclando los estados ocultos de la red en cada iteración, utilizándolos como entrada de la siguiente. Estas redes se utilizan para analizar secuencias temporales, ya que, al reutilizar datos obtenidos anteriormente, permiten relacionar diferentes instantes de la secuencia. Sin embargo, al entrenar una red de este tipo es posible que los valores retropropagados se reduzcan a cero o a infinito, impidiendo el aprendizaje. Esto se conoce como el problema del desvanecimiento del gradiente.

Se han propuesto diferentes modelos para resolver este problema. Entre ellos se encuentran las redes LSTM y las redes GRU.

- Las redes LSTM (Long Short-Term Memory) mejoran la red recurrente básica añadiendo una célula de memoria a cada nodo de la red. La información contenida en esta célula se utiliza en los cálculos y se actualiza en cada iteración. La existencia de estos valores impide que el gradiente se reduzca a cero en retropropagación, aunque no que tienda a infinito.

- Las redes GRU (Gated Recurrent Unit) reducen el problema del desvanecimiento del gradiente de manera similar a las redes LSTM, manteniendo información entre nodos de la red. A diferencia de LSTM, GRU mantiene esta información mediante una serie de puertas que modifican el estado oculto del nodo, añadiendo y substrayendo datos según su relevancia.

Otra variante de las RNN (aplicable a todos los anteriores tipos) son las redes recurrentes bidireccionales, que implementan una segunda capa que procesa los elementos de la secuencia en dirección contraria, permitiendo dar contexto a elementos de la secuencia a partir de segmentos posteriores. Los resultados de ambas capas se combinan para obtener los resultados finales.

2.1.2.3 Redes Convolucionales Recurrentes (CRNN)

El modelo de redes convolucionales recurrentes combina las características de las redes convolucionales y recurrentes. Se compone de una red convolucional con capas de convolución y agrupación, pero reemplaza las capas completamente conectadas con una red recurrente.

2.2 Modelos Encoder-decoder

Los modelos Encoder-Decoder son un diseño de redes neuronales para problemas de mapeo de secuencias a secuencias (como la traducción automática de texto a texto) que se componen de dos partes, un codificador y un decodificador. El codificador reduce la entrada de la red a una representación numérica, resumiendo toda la información a una forma fácil de procesar, como un vector. El decodificador procesa utiliza esta información para crear una secuencia de datos en un formato de complejidad similar a la de los datos de entrada.

Esta arquitectura fue propuesta en 2014 [1] por desarrolladores de Google para la creación de redes secuencia a secuencia, en las que la entrada de la red y la salida contienen información de complejidad similar. Un ejemplo es las tareas de corrección de gramática, en las que se introduce una frase o palabra con errores gramaticales, y se recibe la misma frase con estos errores corregidos. Estos modelos también son útiles para traducción, donde el objetivo es convertir una frase en otra con el mismo significado, pero en un idioma diferente. También es posible utilizar estos modelos para realizar conversiones entre secuencias de complejidad diferente, como audio a texto o texto a audio, aunque en estos casos es conveniente hace algunas adaptaciones.

2.3 Datasets

A la hora de obtener datos para utilización en este tipo de problemas, es importante que las fuentes de audio sean reales, para asegurarnos de que los modelos entrenados tienen aplicaciones reales, y que estén bien etiquetados. Esto supone varios problemas, ya que no es fácil obtener audio en situaciones reales. Además, etiquetar correctamente una cantidad significativa de sonidos conlleva una gran carga de trabajo.

Es por esto por lo que muchos de los conjuntos de datos creados para este campo se componen de sonidos obtenidos de fuentes públicas, como YouTube, o son donados por una comunidad de usuarios.

2.3.1.1 FREESOUND

Freesound [2] es una base de datos colaborativa que contiene archivos de audio de todo tipo recogidos bajo la licencia Creative Commons. Todos los sonidos presentes en Freesound han sido contribuidos por usuarios.

2.3.1.2 Urban Taxonomy

El objetivo del conjunto de datos Urban Taxonomy [3] es crear un repositorio de audio urbano y un sistema taxonómico de clasificación para el mismo que pueda ser utilizado en el futuro. Para crearlo se recogieron sonidos de Freesound, que se etiquetaron manualmente siguiendo el sistema previamente diseñado, especificando la fuente de los sonidos lo más posible. Por ejemplo, se distingue entre los sonidos producidos por el motor de un coche o por sus frenos.

2.3.1.3 Audio Set

La base de datos de audio Google Audio Set publicada en 2017 [4] se compone de fragmentos de audio de diez segundos extraídos de videos de YouTube. Este conjunto de datos tiene 2,084,320 muestras de audio de 527 clases distintas.

2.3.1.4 SINS

El conjunto de datos SINS [5] fue creado a partir de grabaciones tomadas en una casa de cinco habitaciones a lo largo de una semana, durante la cual una sola persona habitó la casa. En total contiene de 1095 fragmentos de audio etiquetados en 16 clases y divididos según en cuál de las cinco habitaciones se produjeron. Fue diseñado para su uso entrenando sistemas para hogares inteligentes.

2.3.1.5 MUSAN

Este conjunto de datos [6] se compone de grabaciones de habla humana, música, y ruido extraídas del dominio público y de fuentes con licencias Creative Commons, tales como audiolibros, charlas del congreso de EE. UU. y música clásica. Estas grabaciones están divididas según su tipo y la fuente de la que se obtuvieron.

2.3.1.6 DESED

El conjunto de datos DESED (Domestic Environment Sound Event Detection Dataset) [7] diseñado para ser usado en problemas de detección de eventos de audio. Dado que es el conjunto de datos sobre el que se va a trabajar en este trabajo, se describirá con más detalles. Se compone de fragmentos de audio en entornos domésticos, extraídos de las siguientes fuentes:

- Grabaciones reales extraídas del conjunto de datos AudioSet
    - 14412 archivos no etiquetados.
    - 1578 archivos débilmente etiquetados, destinados a entrenamiento.
    - 1168 archivos fuertemente etiquetados destinados a validación.
    - 628 archivos extraídos de YouTube.
- Sonidos sintéticos generados automáticamente. Estos sonidos se generan combinado sonidos de fondo y de primer plano extraídos de diferentes bases de datos mediante la librería de python Scaper
    - Los sonidos utilizados como fondo son extraídos de los conjuntos de datos SINS y MUSAN, y de YouTube.
    - Los sonidos utilizados como primer plano son una selección de archivos de la base datos FreeSound, editados para eliminar silencios.

2.4 Mecanismos de Atención

Los mecanismos de atención son un método utilizado en aprendizaje automático ideado para el reconocimiento de objetos en imágenes, permitiendo detectar las áreas que tienen mayor probabilidad de ser relevantes. Esta técnica se ha empezado a usar más recientemente en modelos secuencia a secuencia.

El funcionamiento de los mecanismos de atención para análisis de secuencias consiste en, a partir de dos secuencias de datos, generar una matriz que representa la relación entre los elementos de ambas. También se puede usar la atención para encontrar las relaciones entre diferentes elementos de una misma secuencia, en un proceso que se conoce como auto-atención.

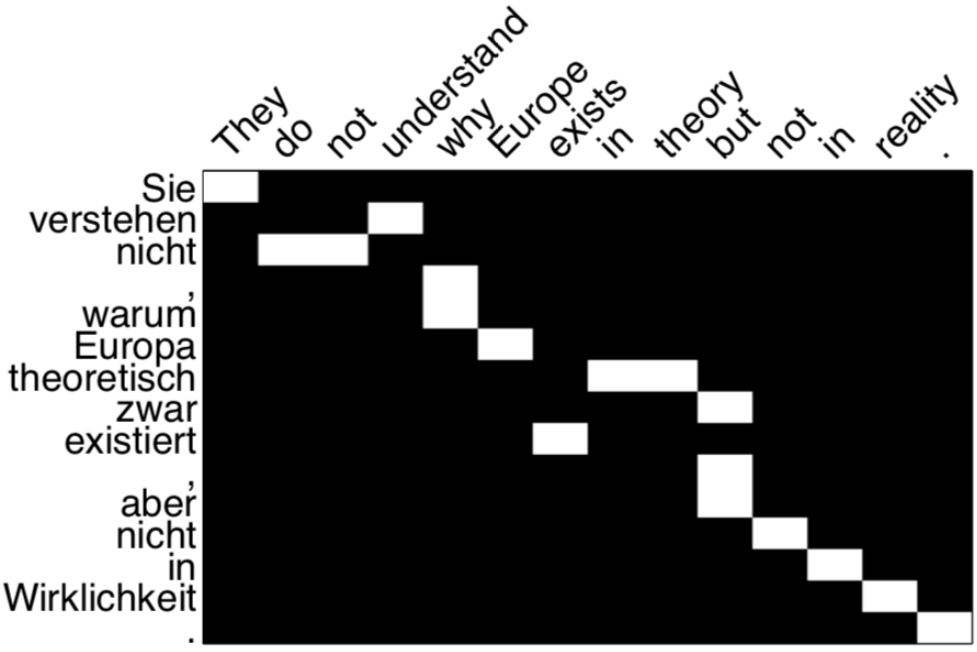

Ilustración 1: Visualización de una matriz de atención [15]

La atención permite analizar la relación entre un elemento de una secuencia de datos y su contexto. Se ha utilizado para obtener buenos resultados en campo, como la traducción de texto, el reconocimiento de voz y el análisis de audio.

Internamente, las funciones de atención realizan un mapeo de un vector de consultas y un conjunto de pares clave-valor en forma de vectores a un vector de salida. Esta salida se calcula como la suma ponderada de los valores, donde los pesos se obtienen aplicando una función de compatibilidad entre las consultas y las claves.

En el caso de la autoatención, los vectores de consultas y claves son iguales al vector de valores. El resultado de esta operación representa la relación entre las diferentes partes de la secuencia de los valores.

La atención multicabeza consiste en realizar la operación de atención varias veces en paralelo, una por cada cabeza, concatenar los resultados y proyectarlos a la dimensión original mediante una capa lineal. En la práctica se realiza una sola operación sobre los valores proyectados a su dimensión original multiplicada por el número de cabezas.

2.4.1 Atención por producto escalar

La función de atención utilizada en el Transformer y en el Conformer es la atención por producto escalar. En este tipo de atención, se realiza la función de compatibilidad producto escalar de las consultas y las claves y se realiza la función softmax sobre el resultado para generar el vector de pesos que se aplica a los valores. En la práctica, se realiza esta operación varias veces de forma simultánea agrupando varios conjuntos de consultas, claves y valores en matrices. La versión del mecanismo de atención por producto escalar utilizada en la red transformer aplica un factor de escala al producto de las consultas y claves, dividiendo cada elemento del vector por la raíz cuadrada de su longitud para evitar posibles errores en la función softmax debido a gradientes extremadamente pequeños.

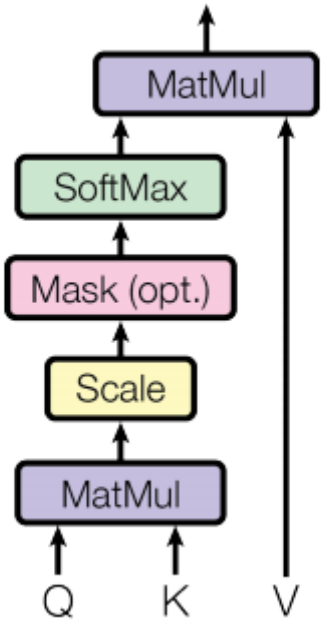

Ilustración 2: Atención por producto escalar [9]

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Ecuación 1: Ecuación de la atención por producto escalar [9]

En esta fórmula, Q es el vector de consultas, K es el vector de claves, V es el vector de valores y d_k es la longitud del vector de claves.

2.5 La red transformer

La red transformer fue introducida en el artículo *Attention is all you need* [9], publicado en 2017. Es un modelo Encoder-Decoder diseñado originalmente para su uso en problemas de traducción de texto. Se basa en modelos anteriores con codificadores-decodificadores basados en redes recurrentes y convolucionales, conectados por una capa de atención, con la idea de incluir la atención en la propia estructura del modelo Encoder-Decoder.

El diseño de la red Transformer tiene como objetivo aplicar el proceso de atención a las secuencias de datos, para extraer información de las relaciones entre las diferentes partes de esta. Tiene una estructura codificador-decodificador, cada uno compuestos de una serie de capas idénticas.

Cada capa del codificador Transformer se compone de dos subcapas, una de autoatención multicabeza y otra lineal *feed-forward*.

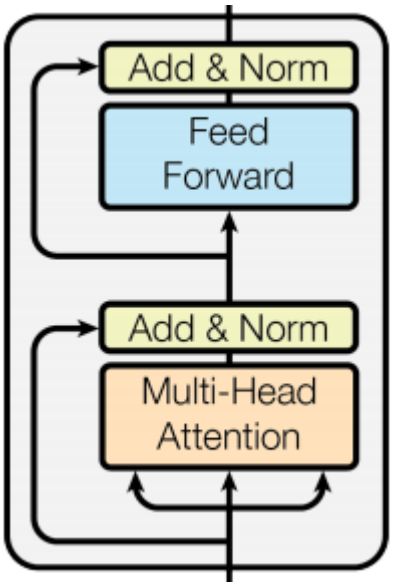

Ilustración 3: Bloque encoder [9]

La capa de autoatención implementa la atención por producto escalar multicabeza, explicada en el apartado 2.4.

La segunda subcapa es una red *feed-forward* por posición, implementada como dos capas lineales completamente conectadas con una capa ReLU entre ambas.

Ambas capas tienen conexiones residuales, es decir, suman su salida a su entrada y normalizan, para mitigar la degradación de la información durante la ejecución de la red.

Las capas del decodificador son similares a las del codificador, pero añade otra subcapa de autoatención al principio. Esta subcapa aplica una máscara a la entrada para forzar al decodificador a predecir el siguiente elemento de la secuencia de salida sólo a partir de los elementos anteriores, sin tener acceso a dicho elemento ni a los elementos futuros. Además, la segunda subcapa de autoatención recibe como claves y valores la salida del codificador. Al igual que las otras dos capas, esta capa implementa conexiones residuales con normalización.

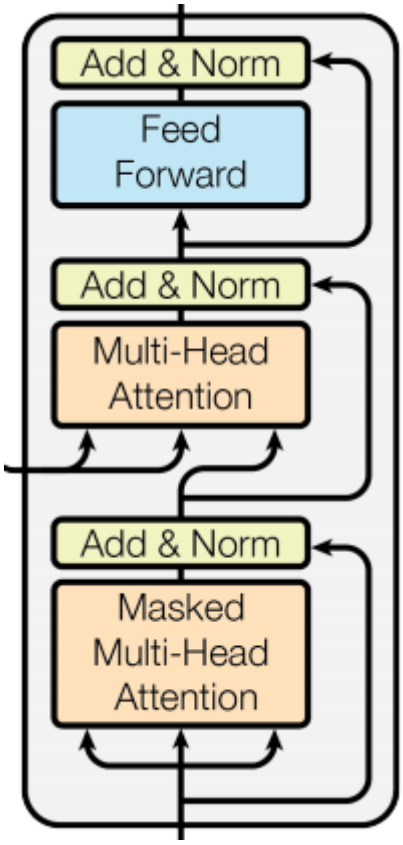

Ilustración 4: Bloque decoder [9]

El modelo original de la red Transformer propuesto en *Attention is all you need* [9] utiliza tanto el decodificador como el decodificador en pilas de seis capas.

Durante el entrenamiento, la frase que se va a traducir se introduce a la pila de codificadores tras procesarla para producir un vector que contiene información posicional. El resultado de este proceso es una serie de vectores que representan la relación entre las palabras de la frase.

A continuación, se introduce a la pila de decodificadores la frase en el idioma objetivo, con información posicional igual que la calculada en la pila de codificadores. La capa de atención con máscara crea un vector de atención que excluye la relación entre cada palabra y ella misma y las palabras posteriores. A continuación, se calcula la atención entre esta matriz y la matriz de atención obtenida de la pila de codificadores. Este proceso se repite en cada decodificador.

Una vez obtenida la salida del decodificador, se introduce a una red lineal simple y se ejecuta la operación softmax. Esto calcula, para cada palabra, la probabilidad de la siguiente palabra de la frase. Con esta información se realiza el entrenamiento.

2.6 La red conformer

El artículo *Conformer: Convolution-augmented Transformer for Speech Recognition* [10], publicado en el 2020 por Google, introdujo la red Conformer, diseñada para tareas de reconocimiento de voz. Esta red está diseñada para ser utilizada como codificador en modelos Encoder-Decoder, con otro tipo de red como decodificador.

El concepto de la red Conformer es combinar las ventajas de las redes convolucionales, que identifican la relación entre elementos cercanos de una secuencia de datos, y la autoatención, que permite relacionar cada elemento con el conjunto de la secuencia. La combinación de ambos métodos produce mejores resultados que su uso por separado.

El bloque Conformer se compone de dos módulos principales: una capa de autoatención y una capa de convolución. Estas capas son precedidas y seguidas por dos capas *feed-forward* de medio paso, imitando las mejoras a la red Transformer de la red Macaron-net. Todas las capas suman los datos de entrada a los datos de salida.

El módulo *feed-forward* se compone de una capa de normalización, seguida de una capa lineal con activación Swish que multiplica la dimensión por cuatro y otra capa lineal que la devuelve a su dimensión original. La capa Swish activation sirve para regularizar la red.

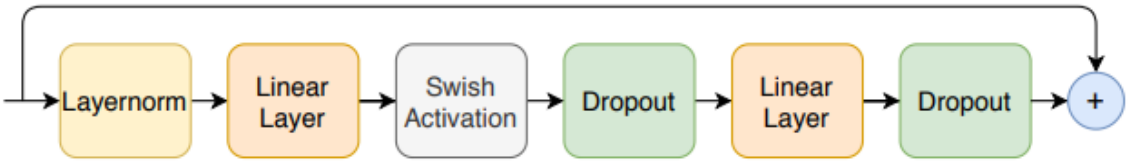

Ilustración 5: Módulo *feed-forward* [10]

El módulo de atención se compone de una capa de normalización, una capa de atención multicabeza, y una capa de dropout. La capa de atención multicabeza es similar a la implementada en el transformer.

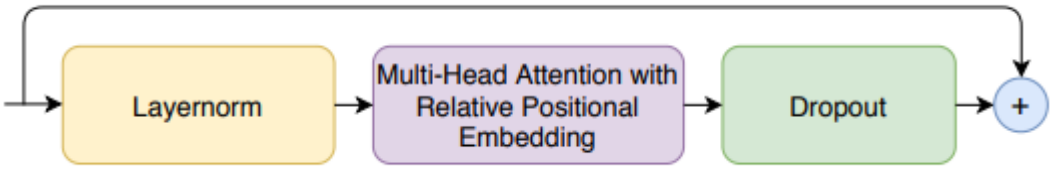

Ilustración 6: Módulo de atención [10]

El módulo de convolución comienza con una capa de convolución que expande la dimensión por un factor de dos. Esta capa es de tipo *Pointwise Convolution*, es decir, tiene un kernel de dimensión 1x1, que itera sobre cada elemento del vector de datos.

La siguiente capa es una capa de activación GLU. Esta capa controla la cantidad de información que avanza a la siguiente etapa de la red, aplicando la siguiente fórmula.

$$h_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$

Ecuación 2: Función de activación GLU [13]

En esta fórmula, X es el vector de entrada, W y V son matrices de pesos, b y c son sesgos, y σ es la función sigmoidal.

Entre estas dos capas se forma un mecanismo de puerta que filtra la información más importante a las capas posteriores.

A continuación, hay una capa de convolución por profundidad de una dimensión. Esta capa divide los datos en varios canales y aplica un filtro de convolución diferente a cada uno de ellos. Tras esta capa hay una capa de normalización *Batchnorm* para estabilizar la ejecución en redes profundas. *Batchnorm* es un mecanismo de normalización que fija tanto la media de los datos a 0, y su varianza a 1.

La siguiente capa es una capa de activación Swish. Esta capa aplica la siguiente fórmula:

$$f(x) = x\sigma(x)$$
$$\sigma(x) = (1 + e^{-x})^{-1}$$

Ecuación 3: Función de activación Swish [14]

En esta ecuación, x es el vector de entrada, y σ es la función sigmoidal.

Esta fórmula tiene como resultado la siguiente gráfica:

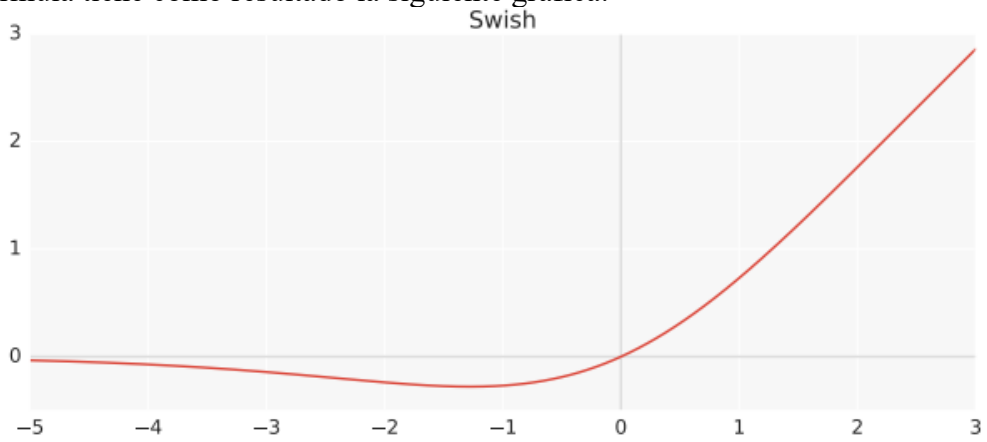

Ilustración 7: Representación gráfica de la función de activación Swish [14]

La ventaja de utilizar esta función de activación sobre otras es que reduce el número de neuronas muertas, es decir, neuronas cuyo valor de gradiente se mantiene a cero.

Finalmente se realiza una segunda convolución que devuelve los datos a su dimensión original.

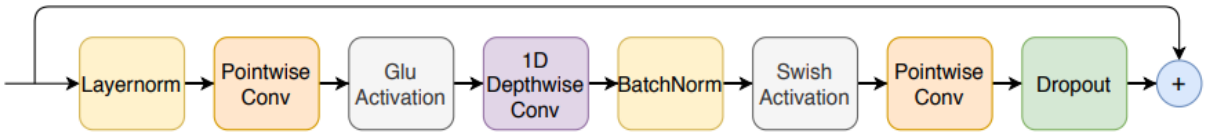

Ilustración 8: Módulo de convolución [10]

3 Diseño

3.1 Baseline de DCASE

Dado que el objetivo de este trabajo de fin de grado es implementar un modelo de red neuronal diseñado para resolver la cuarta tarea del desafío DCASE2020, se ha desarrollado sobre la implementación *baseline* de esta tarea. Esta implementación incluye una red CRNN, y los archivos necesarios para entrenarla sobre el conjunto de datos DCASE. Este entrenamiento emplea el método *Mean Teacher* de entrenamiento semisupervisado.

Para la realización de este trabajo de fin de grado, se han creado versiones modificadas de varios archivos de código, reemplazado el modelo CRNN con los modelos Transformer y Conformer.

Los archivos utilizados implementan las funciones necesarias para entrenar y evaluar modelos de redes neuronales sobre el conjunto de datos DESED. Los modelos deben producir como resultado una predicción débil, que estima una clase para cada archivo de audio, representando el evento de audio principal, y una predicción fuerte, que predice una clase para cada frame del clip de audio, permitiendo la detección de segmentos de audio dentro del archivo.

3.1.1 Modelo

El *baseline* del desafío DCASE2020 es un modelo CRNN, basado en el modelo que obtuvo el segundo puesto del desafío DCASE2019.

La primera etapa del modelo *baseline* es una red convolucional de siete capas. La segunda etapa del modelo es una red recurrente GRU bidireccional. Finalmente, una capa lineal completamente conectada y una capa de activación sigmoidal calculan la probabilidad de cada clase. Estos resultados se utilizan como predicción fuerte. Las predicciones débiles se obtienen a partir de la media de las predicciones fuertes.

3.1.1.1 Capa CNN

Los parametros utilizados en la etapa convolucional son:

- Función de activación GLU (Gated Linear Unit).
- Kernel de tamaño 3x3 para todas las capas.
- Stride de 1 para todas las capas.
- Padding de 1 para todas las capas.
- Número de filtros de cada capa de convolución de 16, 32, 64, 128, 128, 128, 128.
- Tamaño del kernel de cada capa de agrupación de (2, 2), (2, 2), (1, 2), (1, 2), (1, 2), (1, 2), (1, 2).
- Dropout de 0,5.

3.1.1.2 Capa RNN

La capa RNN es una red GRU bidireccional de dos capas, sin dropout.

3.2 Conjunto de datos

Los datos utilizados para la cuarta tarea del desafío DCASE2020 es una combinación de varios conjuntos de datos. La mayoría de los datos proceden del conjunto de datos DESED, repartidos de la siguiente manera:

- Datos reales débilmente etiquetados para entrenamiento. 1578 archivos.
- Datos reales no etiquetados para entrenamiento. 14412 archivos.
- Datos sintéticos para entrenamiento. 2060 archivos de fondo y 1006 archivos principales.
- Datos reales para validación. 1168 archivos.

Los sonidos de fondo para la generación de audio sintético proceden de los conjuntos de datos SINS y TUT Acoustic Scenes 2017**.**

3.3 Lenguaje utilizado

El lenguaje utilizado en el *baseline* de DCASE2020, y por tanto en este proyecto, es python 3.6.

Pytorch es una librería de código abierto para aprendizaje automático creada en Python por el grupo de desarrollo FAIR (Facebook's AI Research lab). Está basada en la librería Torch, programada en Lua. Pytorch implementa un tipo de datos llamado Tensor que almacena matrices multidimensionales de números. Los tensores pueden ser operados en tarjetas gráficas NVIDIA compatibles con la plataforma CUDA, que permite utilizar tarjetas gráficas para procesamiento de datos. Esto permite ejecutar operaciones de forma más eficiente, aprovechando la capacidad de las tarjetas gráficas de realizar un gran número de operaciones en paralelo.

3.4 Medidas

Para medir el éxito del entrenamiento del modelo, se han utilizado las medidas proporcionadas por el *baseline* de DCASE. Esto permite comparar los resultados obtenidos con los del modelo CRNN incluido en el *baseline*.

Las medidas tomadas están divididas entre predicciones de la clase principal del evento de audio, y predicciones del segmento en el que se incluye la clase. A su vez estas predicciones están divididas entre estadísticas generales (utilizando micro-average) y medidas por clase (utilizando macro-average).

Para todas estas estadísticas se toman las medidas F, incluyendo la precisión, la sensitividad, y la medida F1.

También se toman medidas del error de las predicciones. Para las medidas por clase, se toma la tasa de errores, la tasa de borrados y la tasa de inserciones. Para las medidas generales se añade además la tasa de reemplazos.

Para las medidas por segmentos, se añade además medidas de exactitud (accuracy). Concretamente se toman la sensibilidad, especificidad, la exactitud equilibrada y la exactitud.

Finalmente se calcula la puntuación PSDS (Polimorphic Sound Detention Score, explicado en 3.4.2).

3.4.1 F-score

El valor-F o F-score es una medida de la precisión que alcanza un modelo en una prueba de clasificación. Se calcula realizando la media harmónica de la precisión y la sensibilidad. Esta media se calcula como el inverso de la media de los inversos de estos números.

La F-score es útil a la hora de evaluar sistemas de clasificación o búsqueda de información, pero no tiene en cuenta la posibilidad de que la precisión o sensibilidad puedan ser más importantes que la otra en ciertos modelos. Existen variaciones del valor-F que asignan pesos diferentes a estas puntuaciones, pero no son utilizadas en este TFG.

3.4.2 Polyphonic Sound Detection Score (PSDS)

La PSDS es una medida diseñada específicamente para evaluar la detección de eventos polifónicos con el objetivo de complementar el valor-F, cubriendo sus deficiencias. Entre estas deficiencias se encuentra:

- **Único punto de operación:** El valor-F define un único umbral de decisión para cada clase de evento, sin buscar posibles valores óptimos que den pesos más adecuados a la precisión o la sensibilidad. PSDS define varios umbrales y realiza la media de los resultados obtenidos para cada uno de ellos, por lo que los resultados obtenidos son independientes al umbral escogido.

- **Subjetividad del conjunto de datos:** Las anotaciones de eventos de audio en el tiempo tienen un cierto grado de subjetividad. Varias ocurrencias consecutivas de un sonido pueden considerarse como múltiples eventos o como uno solo, y puede ser difícil determinar cuando comienza un sonido y acaba otro. Para evitar los posibles problemas que surgen al evaluar este tipo de casos con diferentes medidas, PSDS basa la comparación de predicciones y etiquetas en las intersecciones entre eventos en lugar de en los periodos de tiempo.

- **Importancia de los cross-triggers:** Los cross-triggers son falsos positivos que corresponden a una clase de evento diferente. Son comunes cuando se tratan tipos de evento de audio similares, por lo que pueden indicar un sesgo en los datos en lugar de un fallo del modelo.

Para calcular la medida PSDS se utilizan dos criterios para calcular los aciertos y los falsos positivos.

- **Detection Tolerance Criterion (DTC):** El criterio de tolerancia de detección escoge una intersección mínima entre las predicciones y las etiquetas en el tiempo. Las predicciones que no coincidan con las etiquetas por encima de este mínimo se consideran falsos positivos.

- **Ground Truth tolerance Criterion (GTC):** Este criterio controla el porcentaje mínimo de una etiqueta en el tiempo que debe coincidir con las predicciones para considerarse un acierto. Las predicciones que cumplen este criterio y el anterior se consideran verdaderos positivos.

- **Cross-trigger Tolerance Criterion (CTTC):** Este tercer criterio determina el porcentaje mínimo de una etiqueta en el tiempo que debe estar cubierto por las predicciones para considerarse un cross-trigger.

Todos estos criterios tienen un parámetro ρ que define su ratio de intersección. En la tarea DCASE2020 Task 4, estos parámetros son ρ = 0,5 para DTC, ρ = 0,5 para GTC y ρ = 0,3 para CTTC. El parámetro αct define el coste de los cross-triggers, y αst penaliza la inestabilidad en las tasas de aciertos de diferentes clases. Esto da lugar a tres medidas. PSDS básica, donde tanto αct como αst son iguales a 0, PSDS Cross-Trigger, donde αct es igual a 1 y αst es igual a 0, y PSDS Macro, donde αct es igual a 0 y αst es igual a 1.

4 Desarrollo

El objetivo de este trabajo es programar el modelo ganador de la cuarta tarea de DCASE2020, explicado más adelante, incluyendo los módulos transformer y conformer. Existen varias implementaciones de la red transformer, y una de estas se ha utilizado como base para implementar esa variación de la red. Por otro lado, no existe ninguna implementación de la red conformer disponible para su libre uso aparte del módulo de convolución, por lo que ha sido necesario implementarla a partir de la red transformer.

4.1 Estructura del modelo

El modelo implementado tiene una estructura lineal, en la que los datos son procesados por varios codificadores basados en modelos de atención, y clasificados por un clasificador por posición.

La estructura se compone de un extractor de características, seguido de un modelo de atención, y finalmente una red *feed-forward* por posición. Tras la ejecución del extractor de características

4.1.1 Extractor de características

El primer módulo del diseño es una red convolucional que prepara los datos para su procesamiento. Esta red es idéntica a la del modelo CRNN *baseline*.

4.1.2 Token

Al resultado del extractor de características se le añade un token de identificación al principio de cada secuencia de audio. Este token se inicializa como un vector unidimensional de longitud igual al tamaño del lote (el número de datos utilizados procesados de una vez durante el entrenamiento), con todos los valores inicializados a cero. A continuación, una capa de Embedding expande la dimensión para crear un vector de tamaño igual a la dimensión de cada frame por cada elemento del lote, permitiendo concatenarlo como un frame extra al principio.

Durante el proceso de atención, los valores del token serán modificados en relación con el total de la secuencia, permitiendo obtener información a partir de su estado final. El resultado de este token se utiliza para la predicción débil.

4.1.3 Clasificador por posición

El clasificador por posición es una red simple completamente conectada compuesta por dos capas lineales entre las que se sitúa una función ReLU. La primera capa lineal multiplica el tamaño del vector de entrada por cuatro, y la segunda lo reduce a su dimensión original.

Tras la ejecución de este módulo de la red se calculan las probabilidades relativas de cada clase a partir del resultado de los módulos de atención.

4.1.4 Módulos de atención

Los módulos de atención en estas redes son los módulos Transformer y Conformer.

4.1.4.1 Transformer

El módulo transformer incluido es una secuencia de tres codificadores transformer. Las primeras pruebas utilizaban una red transformer completa, incluyendo codificador y decodificador. Sin embargo, tras realizar varios experimentos, se ha comprobado que el decodificador es innecesario, ya que el objetivo de esta red es sintetizar y reducir la información de la secuencia de audio original. Tras reducir la red a una serie de codificadores se observó una gran mejora de los resultados.

Para implementar este modelo se ha utilizado una implementación ya existente proporcionada por el grupo de investigación AUDIAS.

Los parámetros utilizados en este módulo son 16 cabezas de atención, y un tamaño interno de vector de 512.

4.1.4.2 Conformer

El módulo conformer se compone de una secuencia de cuatro decodificadores conformer. Al igual que el módulo transformer, no es necesario añadir un bloque decodificador. Para el módulo de convolución, se utilizó inicialmente parte del código implementado por el usuario de Github Phil Wang [9].

Inicialmente se intentó implementar el módulo conformer de la red sin basarse en la implementación transfomer. Esta versión implementaba el módulo de embeddings con posiciones relativas. Sin embargo, esta versión no produjo resultados útiles, por lo que fue descartada.

Tras esto se realizó una nueva implementación a partir del modelo transformer, reemplazando las capas del codificador transformer con las del codificador conformer.

En este módulo se utilizan 4 cabezas de atención, y un tamaño interno de vector de 144.

4.2 Modificaciones realizadas al baseline

Para implementar los modelos propuestos, se han realizado modificaciones al código del *baseline* que reemplazan el módulo CRNN con las redes implementadas. Estas modificaciones se han separado a archivos separados. Los archivos modificados son:

- **main.py:** El archivo **main.py** ejecuta las acciones necesarias, cargando los datos necesarios para entrenamiento y validación, instanciando el modelo, y realizando las funciones de entrenamiento**.** Se han generado los archivos **main_transformer.py** y **main_conformer.py,** que reemplazan el modelo CRNN con los modelos transformer y conformer respectivamente. Además, se ha añadido funcionalidad para hacer posible iniciar la ejecución a partir de un modelo previamente entrenado. De esta manera se puede interrumpir y continuar la ejecución del programa.
- **TestModel.py:** Es el archivo ejecutado para realizar la evaluación de modelos ya entrenados. También contiene funciones para cargar los parámetros de un modelo previamente entrenado. Estas funciones son utilizadas en los archivos **main** para realizar la evaluación y para cargar modelos para continuar su entrenamiento. Se han generado los archivos **TestModel_Transformer.py** y

TestModel_Conformer.py, que realizan las mismas funciones que el archivo original para los modelos correspondientes.

5 Integración, pruebas y resultados

Para realizar las pruebas se han utilizado los parámetros utilizados por el *baseline*, para realizar una comparación directa. Estos parámetros son un tamaño de lote de 24 y 400 épocas de entrenamiento. Los resultados presentados corresponden a la mejor época de cada ejecución.

5.1 Transformer

La época con mejores resultados es la época 317.

5.1.1 Resultados de la clasificación por eventos

Estos datos representan la eficiencia de la red transformer para clasificar las clases de los eventos de audio presentes en el audio.

5.1.1.1 Medidas generales (micro-average)

La tabla siguiente contiene los datos correspondientes a las estadísticas generales de la red. Es decir, las medidas realizando micro-average de todas las clases. Esta tabla muestra los siguientes datos:

- F-measure (F1): La puntuación F.
- Precision: La precisión. El número porcentaje de resultados positivos que son aciertos.
- Recall: La exhaustividad. El porcentaje de los datos cuya clase se ha acertado.
- Error rate (ER): La tasa de errores.
- Substitution rate: La tasa de reemplazos.
- Deletion rate: La tasa de borrados.
- Insertion rate: La tasa de inserciones.

Estos mismos datos se muestran en las siguientes tablas.

Tabla 1: Puntuación del Transformer en la clasificación por eventos (micro-average)

F-measure	
F-measure (F1)	5.57 %
Precision	15.78 %
Recall	3.52 %
Error rate	
Error rate (ER)	1.14
Substitution rate	0.01
Deletion rate	0.96
Insertion rate	0.18

Estos datos representan la media de las puntuaciones obtenidas por la red trasformer clasificando los eventos de audio. Estos resultados no son muy buenos, pero demuestran que la red es capaz de clasificar.

5.1.1.2 Medidas por clase (macro-average)

Estos datos muestran las medidas de la red para cada clase.

La primera tabla contiene las estadísticas de la red utilizando macro-average. Incluye los mismos parámetros que la tabla anterior, menos la tasa de substitución.

Tabla 2: Puntuación del Transformer en la clasificación por eventos (micro-average)

F-measure	
F-measure (F1)	10.99 %
Precision	16.73 %
Recall	12.76 %
Error rate	
Error rate (ER)	1.41
Deletion rate	0.87
Insertion rate	0.54

La siguiente table contiene estadísticas para cada clase. Estas estadísticas son:

- Nref: El número de datos correspondientes a la clase.
- Nsys: El número de datos clasificados como pertenecientes a la clase.
- F: El valor F.
- Pre: La precisión.
- Rec: La exhaustividad.
- ER: La tasa de errores.
- Del: La tasa de borrados.
- Ins: La tasa de inserciones.

Tabla 3: Puntuación del Transformer en la clasificación por eventos para cada clase

Event Label	Nref	Nsys	F	Pre	Rec	ER	Del	Ins
Blender	96	15	1.8%	6.7%	1.0%	1.14	0.99	0.15
Alarm Bell	420	136	4.3%	8.8%	2.9%	1.27	0.97	0.30
Speech	1754	105	2.2%	19.0%	1.1%	1.04	0.99	0.05
Running Water	237	174	6.8%	8.0%	5.9%	1.62	0.94	0.68
Vacuum Cleaner	92	86	30.3%	31.4%	29.3%	1.35	0.71	0.64
Frying	94	349	23.0%	14.6%	54.3%	3.63	0.46	3.17
Dishes	567	12	0.0%	0.0%	0.0%	1.02	1.00	0.02
Dog	570	6	0.3%	16.7%	0.2%	1.01	1.00	0.01
Cat	341	21	1.1%	9.5%	0.6%	1.05	0.99	0.06
Electric Shaver	65	40	40.0%	52.5%	32.3%	0.97	0.68	0.29

Estos resultados muestran las capacidades del transformer para clasificar cada una de las clases de audio de forma individual. Como podemos comprobar, aunque es capaz de

detectar varias clases con cierto grado de acierto, la mayoría de las clases tienen un nivel de acierto relativamente bajo.

5.1.2 Resultados de la clasificación por segmentos

Estos datos representan la eficiencia de la red transformer para clasificar los eventos de audio presentes en el audio en su contexto temporal. Es decir, que parte del audio corresponde a la clase.

5.1.2.1 Medidas generales (micro-average)

La siguiente tabla muestra las estadísticas realizando micro-average. A los datos presentes en las tablas anteriores se añaden los siguientes:

- Sensitivity: La sensibilidad. Igual a la exhaustividad.
- Specificity: La especificidad. El porcentaje de datos no pertenecientes a una clase que no se clasifican como esa clase.
- Balanced accuracy: La exactitud equilibrada, calculada como la media de la la media dde la sensibilidad y la especificidad.
- Accuracy: La exactitud, calculada como el total de aciertos sobre el total de datos.

Tabla 4: Puntuación del Transformer en la clasificación por segmentos (micro-average)

F-measure	
F-measure (F1)	36.54 %
Precision	60.46 %
Recall	26.18 %
Error rate	
Error rate (ER)	0.79
Substitution rate	0.12
Deletion rate	0.62
Insertion rate	0.05
Accuracy	
Sensitivity	26.18 %
Specificity	98.02 %
Balanced accuracy	62.10 %
Accuracy	90.57 %

Se puede observar que estos datos son mejores que los obtenidos en la clasificación por clase. Los posibles motivos se discuten más adelante. Se puede observar que los valores de la exactitud y la especificidad no son muy útiles, debido a que, para cada clase, las otras nueve son negativos, por lo que la mayoría de las predicciones negativas serán aciertos independientemente de la capacidad del modelo para reconocer la clase correcta.

5.1.2.2 Medidas por clase (macro-average)

Las siguientes tablas contienen las estadísticas de la red para cada clase, realizando macro-average.

Tabla 5: Puntuación del Transformer en la clasificación por segmentos (macro-average)

F-measure	
F-measure (F1)	33.84 %
Precision	68.02 %
Recall	31.77 %
Error rate	
Error rate (ER)	0.91
Deletion rate	0.68
Insertion rate	0.23
Accuracy	
Sensitivity	31.77 %
Specificity	98.06 %
Balanced accuracy	64.92 %
Accuracy	90.57 %

Tabla 6: Puntuación del Transformer en la clasificación por segmentos para cada clase

Event Label	Nref	Nsys	F	Pre	Rec	ER	Del	Ins	Sens	Spec	Bacc	Acc
Blender	538	47	6.8%	42.6%	3.7%	1.01	0.96	0.05	3.7%	99.7%	51.7%	95.1%
Alarm Bell	1060	821	65.9%	75.5%	58.5%	0.60	0.42	0.19	58.5%	98.0%	78.2%	94.2%
Speech	3745	251	11.4%	90.8%	6.1%	0.95	0.94	0.01	6.1%	99.7%	52.9%	68.0%
Running Water	1385	927	57.9%	72.2%	48.3%	0.70	0.52	0.19	48.3%	97.3%	72.8%	91.2%
Vacuum Cleaner	801	498	66.4%	86.5%	53.8%	0.55	0.46	0.08	53.8%	99.3%	76.6%	96.0%
Frying	794	2050	50.0%	34.7%	89.5%	1.79	0.10	1.69	89.5%	99.9%	88.2%	87.1%
Dishes	754	24	3.6%	58.3%	1.9%	0.99	0.98	0.01	1.9%	99.9%	50.9%	93.2%
Dog	1132	17	1.6%	52.9%	0.8%	1.00	0.99	0.01	0.8%	99.9%	50.4%	89.8%
Cat	728	50	9.3%	72.0%	4.9%	0.97	0.95	0.02	4.9%	99.9%	52.4%	93.6%
Electric Shaver	522	277	65.6%	94.6%	50.2%	0.53	0.50	0.03	50.2%	99.9%	75.0%	97.5%

Estos datos muestran la eficacia del transformer a la hora de detectar los segmentos correspondientes a cada evento de audio en el tiempo. Los datos muestran que el transformer detecta los segmentos en el tiempo mejor que los eventos por sí solos. Esto puede deberse a un mal funcionamiento del token de clasificación, que es responsable de la clasificación débil. Esto puede estar afectando también al algoritmo de entrenamiento *Mean Teacher*, produciendo malos resultados en el modelo profesor que afecten al entrenamiento del modelo principal. Algunas clases son detectadas con mucha más precisión que otras. Esto puede deberse a que estas clases sean más fáciles de reconocer para la red.

5.1.3 PSDS score

Tabla 7: Puntuación PSDS del Transformer

F1 score (psds eval)	0.28
PSD score αct = 0, αct = 0	0.44
PSD score αct = 1, αct = 0	0.37
PSD score αct = 0, αct = 1	0.22

5.2 Conformer

La época con mejores resultados es la época 374.

5.2.1 Resultados de la clasificación por eventos

Estos datos representan la eficiencia de la red conformer para clasificar las clases de los eventos de audio presentes en el audio.

5.2.1.1 Medidas generales (micro-average)

Estos datos representan la media de las puntuaciones obtenidas por la red conformer clasificando los eventos de audio.

Tabla 8: Puntuación del Conformer en la clasificación por eventos (micro-average)

F-measure	
F-measure (F1)	26.34 %
Precision	37.40 %
Recall	20.33 %
Error rate	
Error rate (ER)	1.12
Substitution rate	0.01
Deletion rate	0.78
Insertion rate	0.33

Los resultados con la red conformer son claramente mejores que los obtenidos por la red transformer.

5.2.1.2 Medidas por clase (macro-average)

Estos datos muestran las medidas de la red para cada clase.

Tabla 9: Puntuación del Conformer en la clasificación por eventos (macro-average)

F-measure	
F-measure (F1)	16.21 %
Precision	26.76 %
Recall	15.45 %
Error rate	
Error rate (ER)	1.37
Deletion rate	0.85
Insertion rate	0.52

Tabla 10: Puntuación del Conformer en la clasificación por eventos para cada clase

Event Label	Nref	Nsys	F	Pre	Rec	ER	Del	Ins
Blender	96	1	0.0%	0.0%	0.0%	1.01	1.00	0.01
Alarm Bell	420	121	4.1%	9.1%	2.6%	1.24	0.97	0.26
Speech	1754	1105	41.1%	53.1%	33.5%	0.96	0.67	0.30
Running Water	237	180	6.2%	7.2%	5.5%	1.65	0.95	0.70
Vacuum Cleaner	92	161	29.2%	23.0%	40.2%	1.95	0.60	1.35
Frying	94	193	15.3%	11.4%	23.4%	2.59	0.77	1.82
Dishes	567	5	0.7%	40.0%	0.4%	1.00	1.00	0.01
Dog	570	353	18.9%	24.6%	15.3%	1.31	0.85	0.47
Cat	341	176	38.3%	56.2%	29.0%	0.94	0.71	0.23
Electric Shaver	65	7	8.3%	42.9%	4.6%	1.02	0.95	0.06

Al igual que los resultados de la red transformer anteriormente mostrados, la red conformer tiene problemas identificando los datos por clase. Sin embargo, obtiene resultados superiores a los de la red transformer.

5.2.2 Resultados de la clasificación por segmentos

Estos datos representan la eficiencia de la red conformer para clasificar los eventos de audio presentes en el audio en su contexto temporal.

5.2.2.1 Medidas generales (micro-average)

Tabla 11: Puntuación del Conformer en la clasificación por segmentos (micro-average)

F-measure	
F-measure (F1)	58.06 %
Precision	78.65 %
Recall	46.01 %
Error rate	
Error rate (ER)	0.60
Substitution rate	0.06
Deletion rate	0.48
Insertion rate	0.06
Accuracy	
Sensitivity	46.01 %
Specificity	98.55 %
Balanced accuracy	72.28 %
Accuracy	93.10 %

Estos datos representan la media de las puntuaciones obtenidas por la red trasformer clasificando los eventos de audio. Al igual que la red trasnformer, el conformer clasifica

mejor los segmentos de audio que las clases. De nuevo, los resultados son superiores a los del transformer,

5.2.2.2 Medidas por clase (macro-average)

Las siguientes tablas contienen las estadísticas de la red para cada clase, realizando macro-average.

Tabla 12: Puntuación del Conformer en la clasificación por segmentos (macro-average)

F-measure	
F-measure (F1)	42.26 %
Precision	67.39 %
Recall	35.62 %
Error rate	
Error rate (ER)	0.79
Deletion rate	0.64
Insertion rate	0.14
Accuracy	
Sensitivity	35.62 %
Specificity	98.55 %
Balanced accuracy	67.09 %
Accuracy	93.10 %

Tabla 13: Puntuación del Conformer en la clasificación por segmentos para cada clase

Event Label	Nref	Nsys	F	Pre	Rec	ER	Del	Ins	Sens	Spec	Bacc	Acc
Blender	538	4	0.0%	0.0%	0.0%	1.01	1.00	0.01	0.0%	100.0%	50.0%	95.1%
Alarm Bell	1060	681	54.8%	74.4%	43.4%	0.72	0.57	0.15	43.4%	98.4%	70.9%	93.1%
Speech	3745	2468	77.8%	97.9%	64.5%	0.37	0.35	0.01	64.5%	99.3%	81.9%	87.5%
Running Water	1385	978	53.5%	64.6%	45.6%	0.79	0.54	0.25	45.6%	96.4%	71.0%	90.0%
Vacuum Cleaner	801	725	61.7%	65.0%	58.8%	0.73	0.41	0.32	58.8%	97.5%	78.2%	94.7%
Frying	794	723	49.4%	51.9%	47.2%	0.97	0.53	0.44	47.2%	96.6%	71.9%	93.1%
Dishes	754	7	1.1%	57.1%	0.5%	1.00	0.99	0.00	0.5%	100.0%	50.3%	93.2%
Dog	1131	876	63.3%	72.5%	56.1%	0.65	0.44	0.21	56.1%	97.6%	76.9%	93.3%
Cat	728	271	49.0%	90.4%	33.7%	0.70	0.66	0.04	33.7%	99.7%	66.7%	95.4%
Electric Shaver	522	33	11.9%	100.0%	6.3%	0.94	0.94	0.00	6.3%	100.0%	53.2%	95.6%

Estos datos representan la eficacia del conformer a la hora de detectar los segmentos correspondientes a cada evento de audio en el tiempo. Podemos ver que la red conformer es capaz de reconocer estos segmentos mejor que la red transformer, obteniendo resultados pobres para un número menor de clases.

Al igual que el transformer, el conformer es capaz de clasificar ciertas clases mejor que otras, llegando al extremo de no tener ningún acierto en una clase. Es posible que esto se

deba al tiempo de entrenamiento, y que un número mayor de etapas de entrenamiento permita clasificar todas las clases de manera más equitativa, o que alguno de los parámetros utilizados sea el que cause estos problemas a la red.

5.2.3 PSDS score

Tabla 14: Puntuación PSDS del Conformer

F1 score (psds eval)	0.39
PSD score αct = 0, αct = 0	0.42
PSD score αct = 1, αct = 0	0.36
PSD score αct = 0, αct = 1	0.23

5.3 Resultados del baseline

Estos resultados se presentan para su posterior comparación con las redes implementadas.

La época con mejores resultados es la época 221.

5.3.1 Resultados de la clasificación por eventos

5.3.1.1 Medidas generales (micro-average)

Tabla 15: Puntuación del modelo baseline para la clasificación por eventos (micro-average)

F-measure	
F-measure (F1)	39.17 %
Precision	42.60 %
Recall	36.26 %
Error rate	
Error rate (ER)	1.10
Substitution rate	0.02
Deletion rate	0.61
Insertion rate	0.47

5.3.1.2 Medidas por clase (macro-average)

Tabla 16: Puntuación del modelo baseline para la clasificación por eventos (macro-average)

F-measure	
F-measure (F1)	34.05 %
Precision	34.33 %
Recall	36.76 %
Error rate	
Error rate (ER)	1.39
Deletion rate	0.63
Insertion rate	0.76

Tabla 17: Puntuación del modelo baseline para la clasificación por eventos para cada clase

Event Label	Nref	Nsys	F	Pre	Rec	ER	Del	Ins
Blender	96	165	38.3%	30.3%	52.1%	1.68	0.48	1.20
Alarm Bell	420	235	24.4%	34.0%	19.0%	1.18	0.81	0.37
Speech	1754	1527	53.6%	57.6%	50.2%	0.87	0.50	0.37
Running Water	237	204	28.1%	30.4%	26.2%	1.34	0.74	0.60
Vacuum Cleaner	92	125	51.6%	44.8%	60.9%	1.14	0.39	0.75
Frying	94	192	28.0%	20.8%	42.6%	2.19	0.57	1.62
Dishes	567	362	23.5%	30.1%	19.2%	1.25	0.81	0.45
Dog	570	404	23.2%	28.0%	19.8%	1.31	0.80	0.51
Cat	341	281	37.9%	42.0%	34.6%	1.13	0.65	0.48
Electric Shaver	65	111	31.8%	25.2%	43.1%	1.85	0.57	1.28

5.3.2 Resultados de la clasificación por segmentos

5.3.2.1 Medidas generales (micro-average)

Tabla 18: Puntuación del modelo baseline para la clasificación por segmentos (micro-average)

F-measure	
F-measure (F1)	74.31 %
Precision	79.68 %
Recall	69.63 %
Error rate	
Error rate (ER)	0.40
Substitution rate	0.08
Deletion rate	0.22
Insertion rate	0.09
Accuracy	
Sensitivity	69.63 %
Specificity	97.94 %
Balanced accuracy	83.78 %
Accuracy	95.00 %

5.3.2.2 Medidas por clase (macro-average)

Tabla 19: Puntuación del modelo baseline para la clasificación por segmentos (macro-average)

F-measure	
F-measure (F1)	68.70 %
Precision	74.26 %
Recall	65.60 %
Error rate	
Error rate (ER)	0.59
Deletion rate	0.34
Insertion rate	0.25
Accuracy	
Sensitivity	65.60 %
Specificity	97.92 %
Balanced accuracy	81.76 %
Accuracy	95.00 %

Tabla 20: Puntuación del modelo baseline para la clasificación por segmentos para cada clase

Event Label	Nref	Nsys	F	Pre	Rec	ER	Del	Ins	Sens	Spec	Bacc	Acc
Blender	538	615	64.9%	60.8%	69.5%	0.75	0.30	0.45	69.5%	97.7%	83.6%	96.3%
Alarm Bell	1060	1009	82.2%	84.2%	80.2%	0.35	0.20	0.15	80.2%	98.4%	89.3%	96.7%
Speech	3745	3324	87.8%	93.4%	82.9%	0.23	0.17	0.06	82.9%	97.0%	89.9%	92.2%
Running Water	1385	835	69.7%	92.7%	55.9%	0.49	0.44	0.04	55.9%	99.4%	77.6%	93.9%
Vacuum Cleaner	801	814	75.7%	75.1%	76.3%	0.49	0.24	0.25	76.3%	98.0%	87.1%	96.4%
Frying	794	948	61.5%	56.5%	67.5%	0.84	0.32	0.52	67.5%	96.0%	81.7%	93.9%
Dishes	754	506	46.5%	57.9%	38.9%	0.89	0.61	0.28	38.9%	97.9%	68.4%	93.9%
Dog	1131	941	66.1%	72.8%	60.6%	0.62	0.39	0.23	60.6%	97.4%	79.0%	93.6%
Cat	728	440	61.8%	82.0%	49.6%	0.61	0.50	0.11	49.6%	99.2%	74.4%	96.0%
Electric Shaver	522	581	70.7%	67.1%	74.7%	0.62	0.25	0.37	74.7%	98.2%	86.4%	97.1%

5.3.1 PSDS score

Tabla 21: Puntuación PSDS del modelo baseline

F1 score (psds eval)	0.59
PSD score αct = 0, αct = 0	0.60
PSD score αct = 1, αct = 0	0.52
PSD score αct = 0, αct = 1	0.44

5.4 Comparación de las medidas del transformer, conformer y baseline

En los datos presentados, se puede observar que el modelo *baseline* obtiene mejores resultados que los modelos desarrollados para este proyecto. Mientras que las redes transformer y conformer suelen tener buenos resultados para varias clases, y malos resultados para otras, la red CRNN del modelo *baseline* consigue clasificar todas las clases de eventos con un alto porcentaje de acierto. Es necesario aclarar que el sistema *baseline* no es un sistema básico, sino que para cada evaluación se suele tomar uno de los mejores sistemas de la edición anterior de la evaluación, de modo que el sistema *baseline* es un sistema con unos resultados ya muy competitivos.

Se han realizado varias pruebas para intentar mejorar los resultados obtenidos, pero ninguna de estas pruebas ha logrado mejorar los resultados. Se han probado diferentes implementaciones del token de clasificación, como eliminar la capa de embedding que amplía su dimensión y generar el vector a un tamaño fijo y cambiar los parámetros de la capa de embedding, pero ninguno de estos cambios ha mejorado los resultados.

Un posible motivo que ya se ha mencionado es que existan problemas con el token de clasificación usado para la predicción débil. Esto puede estar causando problemas a la hora de entrenar el modelo profesor en el algoritmo *Mean Teacher*.

También cabe destacar que no ha sido posible imitar por completo los parámetros de entrenamiento del modelo ganador de la cuarta tarea de la evaluación DCASE2020. Debido a limitaciones de memoria, no ha sido posible utilizar el mismo tamaño de lote, por lo que se ha escogido el utilizado en el modelo baseline. Tampoco ha sido posible realizar el mismo número de etapas de entrenamiento, 30000, debido a que el tiempo necesario para realizar solo 3000 etapas de entrenamiento con el hardware utilizado es cercano al mes, por lo que se ha optado a realizar un entrenamiento con un número de etapas igual al baseline para facilitar la comparación. Esta reducción del número de etapas de entrenamiento es especialmente perjudicial para los sistemas con mecanismos de atención ya que el entrenamiento de los modelos de atención suele requerir un entrenamiento más extenso hasta que los mecanismos de atención aprenden a focalizar la atención correctamente. Es posible que con un tiempo más largo de entrenamiento estas redes alcancen resultados más cercanos a los esperados, pero no ha sido posible abordar esto en este trabajo.

Lo que se puede sacar en claro de estos resultados es que la arquitectura de la red conformer obtiene mejores resultados que la red transformer. Prácticamente todos los parámetros utilizados para evaluar el funcionamiento de las redes son superiores en el modelo conformer. El conformer ha obtenido mejores resultados tanto para la clasificación por clases como para la clasificación por segmentos.

Por otro lado, la puntuación PSDS de los modelos transformer y conformer es similar, aunque la medida PSDS global combina múltiples medidas que se ponderan con distintos pesos, por lo que sería necesario hacer un análisis más detallado de los factores que están favoreciendo a cada modelo con esta medida.

Finalmente, cabe destacar que tras numerosos intentos infructuosos en los que las redes desarrolladas (tanto la transformer, como la conformer) ni siquiera eran capaces de aprender (lo que se observaba en la evolución del coste de entrenamiento) se ha

conseguido desarrollar una red transformer y una conformer que aprenden y que posiblemente consigan resultados mejores cuando se disponga de los recursos hardware necesarios para realizar los entrenamientos completos en un tiempo razonable.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

La realización de este Trabajo de Fin de Grado ha permitido comprobar el funcionamiento de los mecanismos de atención en el campo de la detección de eventos de audio.

Si bien se ha visto que estas redes son eficaces, no se ha podido alcanzar los resultados esperados. Una de las posibles razones de estos resultados es las dificultades para replicar los parámetros de entrenamiento ideales (especialmente el número de etapas de entrenamiento, que se ha tenido que reducir a la décima parte de las empleadas en el sistema que se pretendía emular), debido principalmente a limitaciones del hardware utilizado. También es posible que las implementaciones realizadas puedan ser mejorables en algunos aspectos, destacando la implementación del token para la predicción débil. En cualquier caso, se han conseguido implementaciones de las redes transformer y conformer que entrenan, aprenden y consiguen resultados que, si bien no han conseguido superar al *baseline*, representan un primer paso hacia una optimización que alcance los resultados deseados.

También se ha podido comprobar que el modelo conformer es una mejora clara sobre el modelo transformer, al menos en el campo de la clasificación de eventos de audio.

Como conclusión, se ha podido comprobar que el campo de la detección de eventos de audio tiene una complejidad alta, que requiere de técnicas igualmente complejas. El desarrollo de modelos como los modelos de atención permiten nuevas soluciones y aplicaciones en este y otros campos.

6.2 Trabajo futuro

En este trabajo se ha conseguido poner en funcionamiento las redes Transformer y Conformer para clasificación de eventos de audio, pero los resultados obtenidos son inferiores a los obtenidos por las redes originales. Es posible que sea necesario realizar optimizaciones sobre las redes realizadas para mejorar los resultados de la clasificación. Especialmente importante parece ser la optimización del token que se emplea para la clasificación débil de los tipos de eventos acústicos.

Otra posibilidad sería realizar mejoras al código, ya sea realizando cambios a la estructura de las redes, o mejorando los diferentes módulos que las componen. Experimentar con los parámetros de entrenamiento puede mejorar los resultados, ya que no se conocen todos los parámetros utilizados en la implementación original de los modelos utilizados.

Otro aspecto que será necesario abordar en el futuro es extender el entrenamiento de estas redes, pues en los sistemas originales se entrenaban durante 30000 épocas y en este TFG no hemos llegado a entrenarlas con más de 3000 debido al coste computacional que implicaba. También se deben modificar los parámetros de entrenamiento que no se pudieron utilizar debido a las limitaciones del hardware, entrenando los modelos en un equipo más potente.

Referencias

[1] Kyunghyun Cho, Bart van Merrienboer, Calgar Gulcehre, Fethi Bougares, Holger Schwenk, e Y. Bengio, “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”, Proceedings of the Empiricial Methods in Natural Language Processing, 2014.

[2] Frederic Font, Gerard Roma y Xavier Serra, “Freesound Technical Demo”, ACM International Conference on Multimedia, 2013.

[3] Justin Salamon, Christopher Jacoby y Juan Pablo Bello, “A Dataset and Taxonomy for Urban Sound Research”, Proceedings of the ACM International Conference on Multimedia, 2014.

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

[5] Gert Dekkers, Steven Lauwereins, Bart Thoen, Mulu Weldegebreal Adhana, Henk Brouckxon, Bertold Van den Bergh, Toon van Waterschoot, Bart Vanrumste, Marian Verhelst y Peter Karsmakers, “The SINS Database for Detection of Daily Activities in a Home Environment Using an Acoustic Sensor Network”, Detection and Classification of Acoustic Scenes and Events, 2017.

[6] David Snider, Guoguo Chen y Daniel Povey, “MUSAN: A Music, Speech, and Noise Corpus”, arXiv e-prints, 2015.

[7] Nicolas Turpault, Romain Serizel, Ankit Shah y Justin Salamon, “Sound event detection indomestic environments with weakly labeled data and soundscape synthesis”, Workshop on Detectionand Classification of Acoustic Scenes and Events, Oct 2019.

[8] Phil Wang, “Implementation of the convolutional module from the Conformer paper, for use in Transformers”, accedido en Abril de 2021, https://github.com/lucidrains/conformer.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser e Illia Polosukhin, “Attention is all you need”, Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017.

[10] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu y Ruoming Pang, “Conformer: Convolution-augmented Transformer for Speech Recognition”, Interspeech 2020.

[11] Andrey Temko, Robert Malkin, Christian Zieger, Dušan Macho, Climent Nadeu y Maurizio Omologo, “CLEAR Evaluation of Acoustic Event Detection and Classification Systems”, International Evaluation Workshop on Classification of Events, Activities and Relationships p. 311-322, 2006.

[12] “Detection and Classification of Acoustic Scenes and Events”, accedido en Abril de 2021, http://dcase.community.

[13] Alvaro Durán Tovar, “GLU: Gated Linear Unit implementation”, accedido en abril de 2021, https://medium.com/deeplearningmadeeasy/glu-gated-linear-unit-21e71cd52081

[14] “Deep Learning: The Swish Activation Function”, accedido en Abril de 2021, https://lazyprogrammer.me/deep-learning-the-swish-activation-function/

[15] “Attentionn mechanisms”, https://polakowo.io/datadocs/docs/deep-learning/attention-mechanisms, accedido en Abril de 2021

Glosario (Castellano)

UAM	Universidad Autónoma de Madrid
EPS	Escuela Politécnica Superior
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
CRNN	Red Neuronal Recurrente Convolucional
LSTM	Memoria a Corto Plazo Larga
GLU	Unidad de Cierre Lineal
PSDS	Puntuación de Detección de Audio Polimórfica
DCASE	Detección y Clasificación de Escenas y Eventos Acústicos

Glosary (English)

UAM	Universidad Autónoma de Madrid
EPS	Escuela Politécnica Superior
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
CRNN	Convolutional Recurrent Neural Network
LSTM	Long Short-Term Memory
GLU	Gated Linear Unit
PSDS	Polymorphic Sound Detection Score
DCASE	Detection and Classification of Acoustic Scenes and Events

Anexos

A Manual de instalación

- Instalación del *baseline*:

Para instalar el *baseline* de la cuarta tarea de DCASE2020, se deben seguir los pasos indicados en la página web correspondiente: *https://github.com/turpaultn/dcase20_task4*.

- Instalación de librerias:

Una vez instalado el *baseline*, es necesario instalar las librerías de Python utilizadas en el proyecto. Estas librerías son la librería *conformer*, que solo incluye el módulo convolucionar de esta red, y la librería *torch_optimizer*, que incluye el optimizador RAdam utilizado para el entrenamiento. Para esto se deben ejecutar los siguientes comandos en el entorno Anaconda creado en el paso anterior:

```
pip install conformer
pip install torch_optimizer
```

- Instalación de los archivos:

Una vez realizados estos pasos, se deben insertan los archivos del proyecto en la carpeta /base