MÁSTER UNIVERSITARIO EN CIENCIA Y TECNOLOGÍA ESPACIAL

TRABAJO FIN DE MÁSTER

ANALISIS DE PRIMEROS SONIDOS EN MARTE

Estudiante	*Ortega Rodriguez, Ivan*
Directores	*Saratxaga Couceiro, Ibon* Hueso Alonso, Ricardo
Departamento	Ingeniería de Comunicaciones Física Aplicada
Curso académico	*2021/2022*

Bilbao, 18 de Septiembre, 2022

RESUMEN TRILINGÜE

RESUMEN

El 18 de febrero de 2021 el *rover Perseverance* aterrizó en Marte, iniciando una misión de exploración del planeta Rojo en la que se recogerán las primeras muestras geológicas de la superficie de Marte para ser traídas a la Tierra dentro de varios años por una nueva misión espacial. Entre sus múltiples instrumentos *Perseverance* dispone de dos micrófonos.

La forma en que suenan muchas cosas en la Tierra es diferente en el Planeta Rojo. Esto se debe a que su atmósfera es unas 100 veces menos densa que la atmósfera terrestre en la superficie y tiene una composición diferente a la nuestra, lo que afecta a la propagación del sonido.

Este trabajo presenta un análisis de los sonidos obtenidos, realizado mediante una *Red Neuronal*. El objetivo es desarrollar un sistema capaz de clasificar los sonidos de Marte de manera autónoma identificando ráfagas de viento y sonidos artificiales, como los emitidos por el vuelo en Marte del pequeño helicóptero *Ingenuity*. Para la clasificación de los diferentes sonidos se ha utilizado el *Audio Spectrogram Transformer* (AST), una red neuronal profunda (Deep Neural Network, DNN) para la clasificación de audio (1). Utilizando datos públicos del micrófono *SCAM MIC* se han definido 9 clases de sonido. Tras entrenar a la red neuronal esta clasifica los sonidos disponibles con un alto grado de precisión.

Palabras Clave: Mars 2020, Supercam, Inteligencia Artificial, Machine Listening, Audioset, Transformer

ABSTRACT

The *Perseverance rover* landed on the Red Planet on February 18, 2021, initiating a Mars exploration mission in which it will collect the first samples to be brought back to Earth from the red planet in several years by a new space mission. Among its many instruments *Perseverance* has two microphones for studying the arrival and landing on Mars. Additionally, it captures sounds of the *rover* in operation, wind and other environmental noises.

The way many things sound on Earth is slightly different on the Red Planet. This is because its atmosphere is about 100 times less dense than Earth's atmosphere at the surface and has a different composition than ours, which affects sound propagation.

This paper presents an analysis of the sounds obtained, carried out using a *Neural Network*. The objective is to develop a system capable of autonomously classifying the sounds of Mars by identifying wind gusts and artificial sounds, such as those emitted by the flight of the small *Ingenuity* helicopter on Mars. The *Audio Spectrogram Transformer* (AST), a Deep Neural Network (DNN) for audio classification, has been used to classify the different sounds (1). Using public data from the *SCAM MIC* microphone, 9 sound classes have been defined. After training the neural network it classifies the available sounds with a high degree of accuracy.

Keywords: Mars 2020, Supercam, Artificial Intelligence, Machine Listening, Audioset, Transformer

LABURPENA

2021eko otsailaren 18an, *Perseverance roverra* Planeta Gorrian lurreratu zen, eta Marteko esplorazio-misio bati ekin zion. Misio espazial berri honek planeta gorriaren lehen laginak bilduko ditu, urte batzuk barru Lurrera ekartzeko. *Perseverancek* bere instrumentu ugarien artean bi mikrofono ditu Martera iritsi eta bertan lurreratzeko, roverraren eta haizearen soinuak eta inguruneko beste zarata batzuk aztertzeko.

Planeta Gorrian gauzak ezberdin entzuten dira Lurrarekin alderatuz. Hori gertatzen da bere atmosferak Lurraren atmosferak baino 100 aldiz dentsitate txikiagoa duelako gainazalean eta gureaz bestelako konposizioa duelako, soinuaren transmisioari eragiten diona.

Lan honek lortutako soinuen analisia aurkezten du, *Neurona-Sare* baten bidez egina. Helburua da Marteko soinuak modu automatikoan sailkatzeko gai den sistema bat garatzea, haize-boladak eta soinu artifizialak identifikatuz, eta baita *Ingenuity* helikopterok Marten egindako hegaldiak. Soinu desberdinak sailkatzeko, *Audio Spectrogram Transformer (AST)* erabili da, neurona-sare sakon (Deep Neural Network, DNN) bat (1). *SCAM MIC* mikrofonoaren datu publikoak erabiliz, 9 soinu klase definitu dira. Sare neuronala entrenatu ondoren, honek doitasun handiz sailkatzen ditu eskuragarri dauden soinuak.

Gako-hitzak: Mars 2020, Supercam, Adimen Artifiziala, Machine Listening, Audioset, Transformer

I. ÍNDICE

II. ÍNDICE DE TABLAS

III. ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

1.1. LA MISIÓN MARS 2020: ROVER PERSEVERANCE E INGENUITY

La misión *Mars 2020* (NASA) tiene como objetivo fundamental recoger muestras de la superficie de Marte y almacenarlas en tubos sellados que serán recogidos y traídos a la Tierra por una futura misión espacial, la misión *Mars Sample Return* (2). Además, tiene otros objetivos como caracterizar el clima y la geología de Marte y avanzar las tecnologías necesarias para la futura exploración humana de Marte (3). Entre estos aspectos tecnológicos destacan el helicóptero autónomo *Ingenuity* y el experimento *MOXIE*, capaz de generar oxígeno a partir del CO_2 de la atmósfera marciana.

Se ha elegido el cráter *Jezero* como lugar de aterrizaje para la misión porque se cree que la zona estuvo antaño inundada de agua y albergó un antiguo delta fluvial (4). Por lo tanto, es un lugar adecuado para la búsqueda de indicios pasados de posible vida. Para detectar estos posibles indicios de vida pasada *Perseverance* está recogiendo múltiples muestras de roca y suelo de Marte que serán traídas a la Tierra en un futuro.

Aunque otras misiones espaciales que han viajado a Marte han llevado micrófonos, estos experimentos no habían funcionado con éxito hasta ahora. El rover *Perseverance* cuenta con los micrófonos de aterrizaje (EDL) y el micrófono *SuperCam* (5). El micrófono del sistema EDL se instaló con el objetivo de caracterizar la atmósfera en vertical durante el descenso y aterrizaje de la misión espacial. Desafortunadamente, este micrófono no recogió datos durante el descenso, pero el dispositivo obtuvo sonidos del cráter *Jezero* posteriormente, aunque de manera limitada (6). El micrófono del instrumento *SuperCam* se instaló con el objetivo de complementar las observaciones geológicas de los diferentes espectrógrafos del instrumento *SuperCam*. *SuperCam* dispone de un láser pulsante con el que evaporar pequeños fragmentos de rocas analizando su composición química mediante sus espectrómetros. El sonido del impacto del láser sobre las muestras aporta información adicional sobre estas.

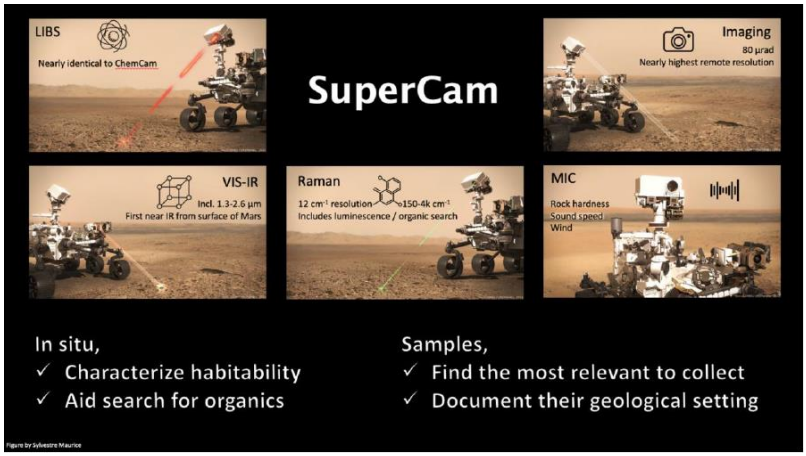

Ilustración 1 – Instrumentos incluidos en *Supercam*

El micrófono de *SuperCam* puede funcionar de manera totalmente autónoma y está aportando una gran cantidad de datos sobre la atmósfera de Marte. Este micrófono puede registrar sonidos durante unos 3,5 minutos seguidos, permitiendo al *rover* escuchar sonidos de Marte, como el sonido de los granos de arena sobre la superficie, el silbido del viento alrededor del mástil y los remolinos de polvo que se acercan al *rover*. El micrófono también graba los sonidos artificiales generados por *Perseverance* cuando utilizando su brazo, perfora rocas para la extracción de muestras o se desplaza por la superficie. Además, también puede escuchar los sonidos generados por los vuelos cercanos del helicóptero *Ingenuity* y estos sonidos pueden utilizarse como ayuda a la hora de a diagnosticar la salud de los mecanismos internos o de los instrumentos del *rover* (5).

1.2. EL SONIDO EN MARTE

El estudio del sonido en una atmósfera extraterrestre como la de Marte puede ofrecer importante información sobre esta. Desde la estructura térmica de la atmósfera cerca de la superficie a diferentes horas del día al grado de turbulencia atmosférica en escalas muy pequeñas, o de estructuras de mayor escala como las ráfagas de viento y vórtices atmosféricos que se pueden encontrar y que generan sonidos al interaccionar con la superficie.

Por todo ello el micrófono de *SuperCam* se activa con regularidad escuchando diferentes fuentes de sonido. Estos sonidos extraterrestres pueden ser muy diferentes en función de las condiciones ambientales y su estudio requiere una buena clasificación de la información disponible.

El viento interaccionando con la superficie del terreno es una fuente de sonidos que requiere un cierto nivel de actividad atmosférica. En una atmósfera fría de CO_2, se espera que la velocidad del sonido sea menor que en la Tierra (7) . Además, debido a la baja presión y a las propiedades físicas del CO_2, también se espera una dispersión de esta velocidad con la frecuencia (8) (9). Sin embargo, la propiedad más notable de la propagación del sonido en Marte es la magnitud de la atenuación en todas las frecuencias, especialmente por encima de 1 kHz (7).

Un primer análisis científico de los sonidos de Marte se encuentra disponible en el artículo *'In situ recording of Mars soundscape'* (7). Esta referencia ha servido de guía fundamental a la hora de desarrollar la clasificación de eventos que se ha utilizado durante este trabajo fin de máster.

Los datos utilizados son las grabaciones de sonido obtenidas por el micrófono de SuperCam. Estos archivos pueden examinarse en formato *pds* (formato del *Planetary Data System* de NASA) y en formato *wav* (en el que pueden escucharse y procesarse como archivos sonoros). Los datos están disponibles de manera pública en el *Planetary Data System* (10) en el sitio web:

https://pds-geosciences.wustl.edu/m2020/urn-nasa-pds-mars2020_supercam/data_calibrated_audio/

En este proyecto, se han examinado sonidos obtenidos por el micrófono de *SuperCam*. Esos sonidos proceden del viento o perturbaciones atmosféricas, del helicóptero, del compresor de instrumento (*MOXIE*) (11), de los mecanismos de rotación del mástil donde se aloja *SuperCam*, del láser y espectrómetros para buscar compuestos orgánicos, de los experimentos realizados con el láser, del polvo golpeando el micrófono y en general del ruido ambiente.

Aunque hasta el momento se han publicado ficheros con sonidos recogidos en Marte hasta el *sol* 418 (un *sol* es un día marciano), en este trabajo se analizan tan solo los sonidos publicados en los primeros 299 soles.

1.3. USO DE REDES NEURONALES PARA EL ANÁLISIS DE LOS SONIDOS EN MARTE

Con el objetivo de desarrollar un sistema de clasificación de los sonidos en Marte registrados por el micrófono de *SuperCam*, se ha optado por el uso de redes neuronales entrenadas mediante aprendizaje supervisado con un conjunto de muestras previamente clasificadas por un ser humano que adjudica una serie de etiquetas a diferentes eventos sonoros. Para etiquetar los diferentes eventos sonoros se han

examinado los espectrogramas de sonido de diferentes archivos comparando con los análisis científicos publicados hasta ahora (7).

En este trabajo se expone como una red neuronal entrenada es capaz de distinguir qué actividad está realizando el *rover* utilizando fragmentos de sonido de solamente 1 segundo de duración con precisiones superiores al 90%. La solución desarrollada está basada una *Red Neuronal* denominada AST (*Audio Spectrogram Transformer*) (12) que utiliza mecanismos de *atención* y *Transformers* junto con transferencia de aprendizaje.

2. OBJETIVOS Y BENEFICIOS

2.1. OBJETIVOS Y ALCANCE DEL TRABAJO

El objetivo del proyecto es estudiar la viabilidad de la utilización de una red neuronal para la clasificación de eventos de sonido recogidos por micrófonos en Marte. Los objetivos específicos del proyecto son:

1. Clasificar los archivos de audio obtenidos en Marte generando una base de datos de los eventos sonoros disponibles en cada archivo.
2. Seleccionar los eventos sonoros de interés.
3. Generar un sistema de clasificación de sonidos basado en redes neuronales que permita detectar y clasificar los eventos de interés.
4. Analizar los resultados para probar el rendimiento de la red con diferentes parámetros.

2.2. BENEFICIOS QUE APORTA EL TRABAJO

Este trabajo desarrolla una herramienta de clasificación de sonidos en Marte que contribuye a la investigación científica de Marte y a su exploración. Los beneficios más directos son los siguientes:

1. Facilitar la detección de los eventos sonoros de interés que permitan comprender mejor para la atmósfera marciana.
2. Crear una herramienta capaz de clasificar los eventos sonoros sobre un conjunto de observaciones que aumenta diariamente.
3. Concentrar las actividades de análisis científico en los eventos de interés, localizando estos rápidamente en la amplia base de datos de las observaciones.
4. Detectar y caracterizar la actividad del *rover* y el helicóptero *Ingenuity*.
5. Facilitar el análisis de futuras observaciones de sonido u otros sonidos dentro de la misma misión.

3. REDES NEURONALES PARA LA CLASIFICACIÓN DE AUDIO

En este capítulo, se describe el concepto de clasificación de sonido, se hace una introducción a las redes neuronales y se profundiza en su aplicación. Después, se describe la red neuronal utilizada.

3.1. CLASIFICACIÓN DE SONIDO

Dado que las señales de audio son interpretadas por el sistema oído / cerebro humano, el mecanismo perceptivo debe ser similar en el software. Este concepto se denomina *machine listening*, donde con el objetivo de rendir al mismo nivel que los humanos, el ordenador escucha y comprende el contenido de forma parecida.

Actualmente, existen una gran variedad de sistemas basados en inteligencia artificial para eliminar la ambigüedad de los sonidos y comprender el entorno acústico. Históricamente, las características hechas a mano y los modelos ocultos de *Markov* (HMM) se usaban para la clasificación de audio (13). Con el auge de las redes neuronales en la última década, *las Redes Neuronales Profundas (DNN)* y especialmente *las convolucionales* (CNN) se han convertido en el bloque de construcción estándar de facto para los modelos de clasificación de audio. Estas herramientas pueden clasificar los sonidos existentes tras un entrenamiento previo y utilizan como entrada normalmente los espectrogramas o incluso directamente las formas de onda de audio para determinar la clase de salida a la que corresponde el sonido.

Más recientemente, se ha propuesto que las redes neuronales basadas puramente en la *autoatención*, como el *transformer* de espectrograma de audio (AST), superan aún más los modelos de aprendizaje profundo construidos con *redes neuronales convolucionales* para aplicaciones de audio (13). En términos sencillos, el mecanismo de *autoatención* es una arquitectura neuronal que permite que el sistema evalúe los datos de entradas y descubran qué partes son más significativas de cara a producir una clasificación correcta de forma que pueda ponderarlas más, es decir, prestarles más atención.

3.2. REDES NEURONALES

Las redes neuronales artificiales son un conjunto heterogéneo y poco delimitado de técnicas matemáticas que se desarrollaron inicialmente durante la segunda mitad del

siglo XX. La palabra neuronal indica que las técnicas tienen algunas similitudes con la forma en que creemos que las neuronas reales procesan la información (14).

Una red neuronal está compuesta por un conjunto de elementos de procesamiento muy simples denominados neuronas. Cada neurona puede recibir múltiples datos de entrada y genera un valor de salida (activación) enviado como señal a una o más elementos de la red. Esta señal normalmente está normalizada para tener valores comprendidos entre cero y uno. Las señales enviadas serán a su vez de los valores de entrada para las demás neuronas de la red. Las neuronas se ordenan en diferentes capas que reciben valores señales de la capa anterior y envían un resultado a una capa posterior. Estos valores se envían mediante conexiones entre neuronas similares a los axones de las neuronas biológicas. Cada neurona recibe entonces las señales de sus compañeras y la reescala mediante los llamados *pesos de conexión* de cada entrada que llega a una neurona desde otra (15).

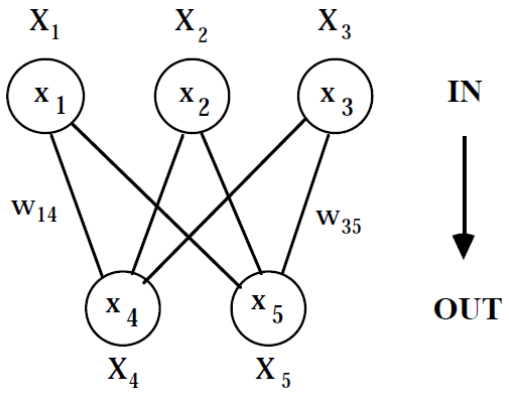

Ilustración 2 – Esquema de una red neuronal prealimentada de una capa *(14)*

La mayoría de las *redes neuronales* tienen varias capas de neuronas y cuando el número de capas es mayor que 3 se dice que son redes neuronales profundas o Deep Neural Networks (DNN). Al añadir más capas y más unidades dentro de una capa, se puede representar funciones de complejidad creciente y de esta manera se definen las *Redes de Aprendizaje Profundo* modernas. Estas *redes neuronales* se utilizan en diferentes campos, pero uno de sus campos de desarrollo clásico es la clasificación de conjuntos de datos como imágenes o sonidos. Las tareas que consisten en asignar un vector de entrada a un vector de salida pueden llevarse a cabo mediante este tipo de redes, siempre y cuando se disponga de modelos y de conjuntos de datos etiquetados suficientemente grandes (16). Las *Redes Neuronales Convolucionales* y los *Transformers* son ejemplos de *Redes de Aprendizaje Profundo.*

Con la llegada de las redes neuronales de aprendizaje profundo, la investigación en clasificación de sonido ha evolucionado de los modelos basados en características específicas de los datos con reglas de clasificación diseñadas de manera casi artesanal (17) (18), a modelos completos generales que pueden utilizarse sobre conjuntos de archivos sonoros muy diferentes con un entrenamiento adecuado para cada tarea de clasificación. Generalmente, estos modelos más avanzados examinan los espectrogramas de audio y las etiquetas introducidas por un usuario humano durante el proceso de aprendizaje (19) (20) (21). En este ámbito de clasificación de audio, las redes neuronales convolucionales (*CNN*) (22), han sido ampliamente utilizadas para aprender representaciones a partir de espectrogramas para el modelado de ficheros de sonido. Las redes convolucionales utilizan capas específicas de neuronas que se especializan en reconocer patrones locales dentro del conjunto de los datos de la entrada. Esto se realiza mediante los llamados filtros convolucionales, que recorren los datos de la entrada para buscar estos patrones, lo que se realiza mediante la operación matemática llamada convolución (16).

Para captar mejor el contexto global alrededor de los sonidos a clasificar, una tendencia reciente es añadir un mecanismo de *autoatención* sobre la red, en los que la clasificación de cada sonido depende no solamente de sus propiedades sino también de su contexto (esta "*autoatención*" es una característica clave en los sistemas de procesamiento de voz en los que la identificación de una palabra puede depender también de su posición dentro de una frase). Matemáticamente, la *autoatención* relaciona numéricamente diferentes posiciones de una misma secuencia para calcular una representación de la misma. Estos con mecanismo de *autoatención* modelos híbridos *CNN*-atención han logrado resultados de vanguardia en muchas tareas de tareas de clasificación de audio, como la clasificación de fenómenos sonoros (23) (24), el reconocimiento de comandos de voz (25) y el reconocimiento de emociones (26).

Dado que los modelos puramente basados en la *atención* en el ámbito de la visión han funcionado correctamente (27) (28) (29), se ha comenzado a prescindir de la *convolución* para la clasificación de sonido, dando lugar a un nuevo tipo de clasificadores llamados *Transformer*. El *Transformer* es una arquitectura de modelo que evita la recurrencia y que se basa totalmente en un mecanismo de *atención*. Además, permite una paralelización significativamente mayor y alcanza mayor calidad en la clasificación de sonido tras un entrenamiento (30).

Por lo tanto, un *Transformer* es un modelo de aprendizaje profundo que adopta el mecanismo de *autoatención*, ponderando diferencialmente la importancia de cada parte de los datos de entrada (31).

3.3. AUDIO SPECTROGRAM TRANSFORMER (AST)

Audio Spectrogram Transformer (AST) es un modelo sin convolución y basado en la *atención* que se aplica directamente a un espectrograma y que puede capturar el contexto global de los sonidos. Además, AST permite transferir conocimiento del *Transformer de Visión (ViT)* (28) preentrenado sobre *ImageNet* (32) a AST, que puede mejorar significativamente el rendimiento del clasificador, especialmente si no hay mucho material de entrenamiento.

Las ventajas de AST son tres. En primer lugar, AST tiene un rendimiento muy elevado. Por ejemplo, se ha evaluado el rendimiento de AST en una variedad de tareas de clasificación de audio y conjuntos de datos como *AudioSet* (33), *ESC-50* (34) y *Speech Commands* (35). En segundo lugar, AST admite entradas sonido de longitud variable y puede aplicarse a diferentes tareas de clasificación de sonidos muy diferentes sin cambios de arquitectura. En concreto, los modelos que utiliza para todas las tareas mencionadas tienen la misma arquitectura mientras que las longitudes de entrada varían desde 1 segundo a 10 segundos, habiendo elegido en nuestro proyecto longitudes de 1 y 2 segundos. En tercer lugar, en comparación con los modelos híbridos *CNN* de atención, *AST* presenta una arquitectura más sencilla, con menos parámetros, y converge más rápidamente durante el entrenamiento.

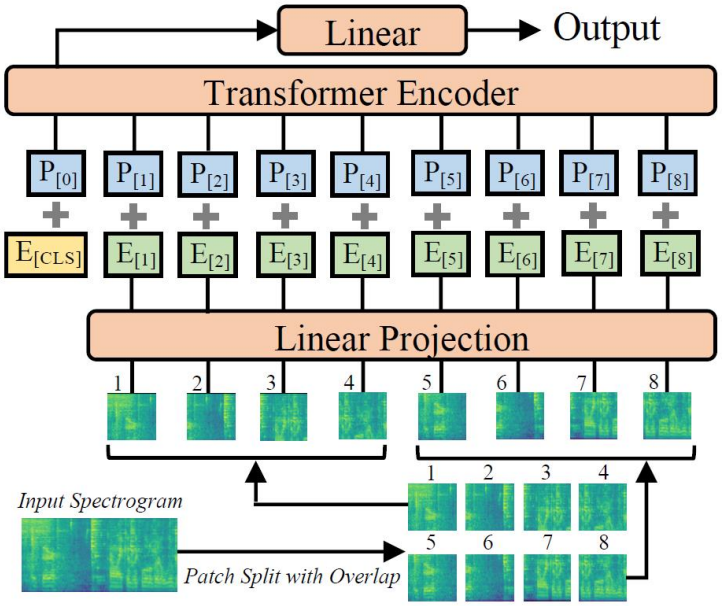

Ilustración 3 – Esquema de *Audio Spectrogram Transformer* (AST) (de la ref. *(1)*)

El sistema *Audio Spectrogram Transformer*, se basa en la arquitectura *Transformer* (30), que se propuso originalmente para tareas de procesamiento del lenguaje natural. Este modelo difiere de otros en que no tiene *convolución* y se basa exclusivamente en mecanismos de *atención*.

4. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

En este apartado se va a describir la solución propuesta. En primer lugar, se describe el formato de los fragmentos publicados en la base de datos de sonidos en Marte (10) y como se han seleccionado algunos ficheros de interés. Después, se hace una descripción de como son los *scripts* para la descarga de los ficheros y la creación de los fragmentos y ficheros de datos para adaptar la información a la Red Neuronal puramente basada en la autoatención, *Audio Spectrogram Transformer* (AST) (1).

4.1. BASE DE DATOS DE EVENTOS DE SONIDO EN MARTE

4.1.1. DATOS PÚBLICOS DE LA MISIÓN MARS 2020

El *nodo de geociencias del Sistema de Datos Planetarios* (*PDS; Planetary Data System*) de la NASA archiva y distribuye datos digitales relacionados con el estudio de las superficies e interiores de los cuerpos planetarios terrestres. Trabaja directamente con las misiones de la NASA para ayudar a generar archivos de datos permanentes y bien documentados. Proporciona datos a los investigadores y asistencia en el uso de los datos. Todos los archivos están en línea y a disposición del público para su descarga gratuita (10).

Se han elegido los ficheros de la misión *Mars 2020 SuperCam Bundle* para la creación de la base de datos. Dentro de este archivo de datos hay diferentes carpetas y se ha elegido la carpeta *data_calibrated_audio* con las grabaciones acústicas calibradas como fuente de datos de audio. Esta carpeta está dividida por información en cada día marciano (*sol*) y dentro se pueden encontrar los ficheros con la información recogida por el sensor. En los nombres de los ficheros se indican valores como el instrumento, el número de segundos del reloj, el tipo de producto, el número de secuencia y el nombre del objetivo, donde se suele incluir si es un evento sonoro de la atmosfera marciana, del compresor, del helicóptero o de la ablación del láser sobre una roca (en estos casos se indica el nombre de la roca). Sin embargo, esta información no siempre está correctamente adscrita y además un mismo archivo sonoro puede contener la superposición de diferentes sonidos en diferentes instantes de tiempo durante la longitud temporal de cada archivo.

El formato de archivo de los archivos de *SuperCam* es *FITS* (Flexible Image Transport System). *FITS* es un estándar de archivo portátil y flexible que permite estructurar los datos de forma eficiente. Se utiliza ampliamente en astronomía y existen varias herramientas que permiten leer su contenido (36). En este trabajo se ha utilizado

QFitsView y la librería de *Python astropy* para gestionar estos ficheros. Cada archivo *fits* puede contener varias particiones divididas en una *cabecera* y una sección de *datos*. La cabecera contiene información sobre el formato de los datos y el número de registros. En estas cabeceras se ha encontrado información utilizada para la conversión a audio de los ficheros *fits*, como la frecuencia de muestreo y ganancia del micrófono.

Los ficheros vienen acompañados de etiquetas donde se describe cada componente del archivo de datos *fits* y que están escritos en *xml* que pueden ser visualizados en un editor de texto o en un navegador *web*. Además, la misma colección de datos incluye ficheros equivalentes para cada archivo *fits* en formato *wav* (audio) con el audio normalizado obtenido en la medida. Esta información se encuentra tratada por el equipo científico del instrumento y se ha utilizado también en los experimentos realizados en el capítulo 'Experimentos y Resultados'.

Entre los metadatos que podemos encontrar en estos ficheros, se encuentran valores como el número de *sol* (día marciano), la hora solar, el instrumento, las coordenadas de tiempo, datos de telemetría como el *azimuth* y la elevación inicial y final, el estado y modo de operación para cada sensor, etc. Se han analizado ficheros de los primeros 299 *soles* por contener una parte mayoritaria de los tipos de eventos de toda la base de datos. Es necesario añadir, que entre los eventos de ruido atmosférico de los *soles* 279 y 299 se han encontrado eventos de interés. Para encontrar más información acerca de lo ocurrido durante la exploración en estos días se puede consular la web '*Analyst's Notebook*' del nodo de geociencias del PDS (37).

4.1.2. BASE DE DATOS DE EVENTOS SONOROS

Se han elegido los ficheros por contar con episodios significativos haciendo un total de 90 ficheros de longitud variable pero típicamente de 30 segundos. Después, se han etiquetado todos los sonidos de interés encontrados en ellos obteniendo 223 eventos. Además, se ha utilizado los fragmentos sin eventos para la clasificación. Estos eventos sonoros se han clasificado a partir de la escucha y análisis de los archivos de audio en formato *wav*, a los que se les ha añadido etiquetas.

Sin embargo, la solución propuesta utiliza los archivos *fits* ya que estos recogen los datos originales sin procesar (36). En el caso del repositorio *data_calibrated_audio* esta información se encuentra en la tabla *Sound*.

Se ha creado un script para automatizar el proceso de extracción y etiquetado de audio de estos ficheros y se puede encontrar información más detallada en el anexo Script *get_input_files.py*. El uso de los archivos *fits* en lugar de los *wav* permite conseguir una clasificación más precisa.

El primer paso ha sido clasificar los eventos encontrados en los diferentes ficheros de audio. Para ello, se han analizado manualmente los audios, escuchándolos y detectando los momentos en la grabación dónde se producen los sonidos de interés. Estos segmentos de la grabación se han marcado mediante etiquetas y se ha creado una tabla Excel con el principio final y duración de cada fragmento.

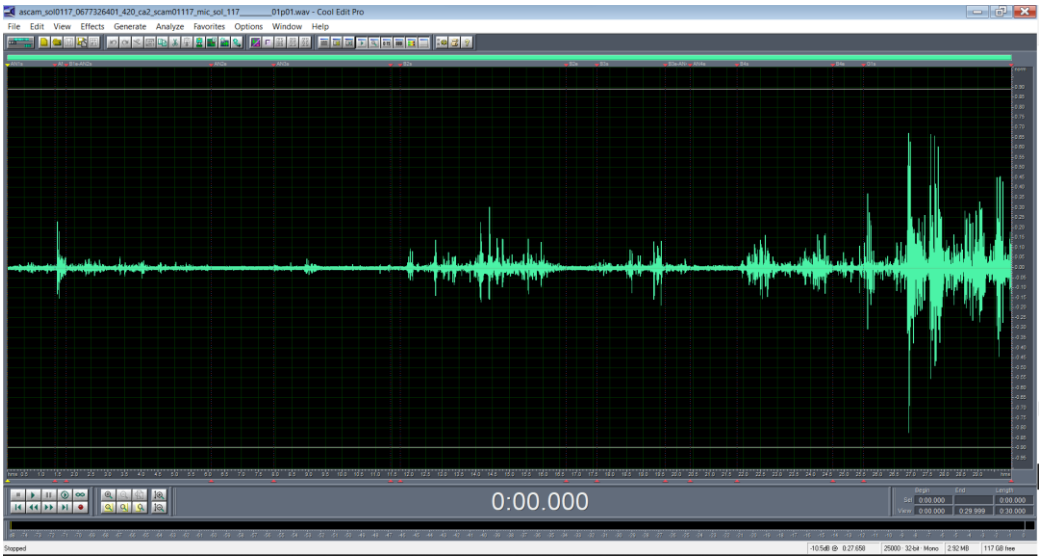

Ilustración 4 – Fichero etiquetado con ruido atmosférico (Cool Edit Pro 2.0)

Se ha creado un documento Excel que contiene una lista de todos los episodios encontrados en una lista determinada de ficheros. Esta lista está contenida en la hoja *Events* y contiene los siguientes campos.

Índice	Ejemplo	Descripción
File name	ascam_sol0117_...wav	Nombre del fichero donde se encuentra el evento.
Dir	sol0117	Día marciano en el que se ha realizado la medida.
Begin	0:00.000	Tiempo con el principio del evento.
End	0:03.462	Tiempo final del evento.
Length	0:03.462	Duración total del evento.
label	B1	*Label* del evento.
description	Breeze	Descripción del evento.
type	B	Tipo de evento en la tabla clases.
mid	breeze	Descripción del evento en la tabla clases.
id	0	Índice del evento en la tabla clases.

Tabla 1 – Descripción de las columnas de la tabla *events*

Las columnas *mid* e *id* se calculan automáticamente mediante una fórmula que busca el tipo de evento en la hoja *Classes* del documento. Esta tabla contiene las clases descritas anteriormente y un recuento de los eventos de cada tipo encontrados.

Una vez realizada esta lista se ha guardado en formato *csv* para ser posteriormente procesada por el script de generación de fragmentos.

Tras realizar esta tarea en los 90 ficheros, se distinguieron inicialmente 9 clases con un total de 3370.36 segundos.

Índice	Descripción	Acrónimo	Nº	Duración (s)	Descripción
0	Breeze	B	43	458.987	Brisa Ligera
1	Gust_of_wind	G	30	295.045	Viento fuerte
2	Atmospheric_noise	AN	57	367.88	Pequeña perturbación
3	Saturated	S	3	44.41	Micrófono saturado por el viento
4	Hélix	H	8	381.124	Ruido presentado por *Ingenuity*
5	Wind+Helix	WH	3	25.905	Viento más helicóptero
6	Moxie	M	19	1112.796	Ruido del compresor *Moxie* (11)
7	Wind+Moxie	WM	9	439.956	*Moxie* con viento
8	Laser	L	51	244.257	Ruido del láser
9	Rest	R	*		Partes no etiquetadas.
		Total	223	3370.36	

Tabla 2 – Diferentes clases encontradas durante la clasificación

Para la clasificación final de los ficheros se ha realizado agrupamiento o *clustering* porque algunos eventos son bastante parecidos entre sí y se ha decidido agruparlos en una sola clase.

Después del agrupamiento, se han distinguido 6 clases.

Clase Origen			Clase Destino			Clases tras el *clustering*	
Índice	Descripción		Índice	Descripción		Índice	Descripción
0	Breeze		0	Breeze		0	Breeze
1	Gust_of_wind		0	Breeze		1	Saturated
2	Atmospheric_noise		0	Breeze		2	Hélix
3	Saturated		3	Saturated		3	Moxie
4	Hélix	→	4	Hélix	→	4	Laser
5	Wind+Helix		4	Hélix		5	Rest
6	Moxie		6	Moxie			
7	Wind+Moxie		6	Moxie			
8	Laser		8	Laser			
9	Rest		9	Rest			

Tabla 3 – Esquema del agrupamiento de clases

Por último, en los espectrogramas de cada uno de los eventos clasificados se encuentran características que permiten clasificarlos dentro de una clase determinada. Aunque estas características no se extienden en el tiempo completo de la muestra se pueden apreciar en alguna de las ventanas. Para las capturas, se han utilizado los

ficheros de datos con audio no normalizado extraídos de los ficheros *fits* ya que los ficheros normalizados cuentan con mucho ruido en algunos casos. Estas características se describen en los siguientes apartados para cada clase final.

4.1.3. DESCRIPCIÓN DE LOS EVENTOS SONOROS DE INTERÉS

Viento (Perturbaciones de diferentes magnitudes)

Esta clase, agrupa el viento fuerte, la brisa y las pequeñas perturbaciones atmosféricas (exceptuando en el Experimento 5: Evento Ruido Atmosférico). A continuación, se muestra una captura de un espectrograma de un fichero de audio recogido durante el *sol 117* donde se pueden ver las características de la clase:

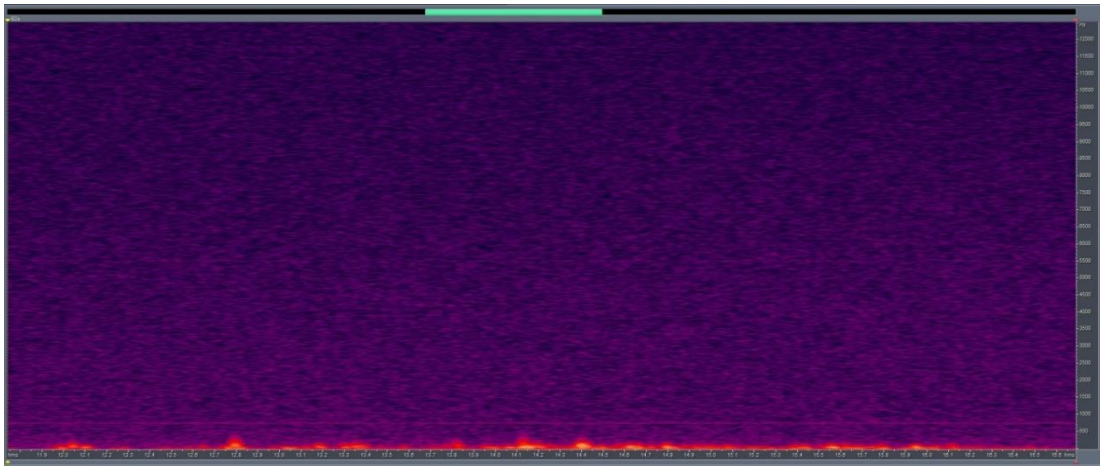

Ilustración 5 – Espectrograma de un evento de la clase Viento

Como se ve en la imagen, aparece máximos (en color amarillo/rojo) en las bajas frecuencias (generalmente por debajo de 500 Hz), quedando el resto del espectro limpio.

Saturación del Micrófono

Algunas perturbaciones son tan fuertes que llegan a saturar la señal recogida por el micrófono. Como resultado, se obtiene un espectrograma con armónicos en todas las frecuencias en algunas ventanas de tiempo, que se puede percibir como líneas rojas verticales con una base de más potencia en las frecuencias bajas (amarilla). A continuación, se muestra un espectrograma de uno de los ficheros del *sol 117* donde se puede ver este fenómeno.

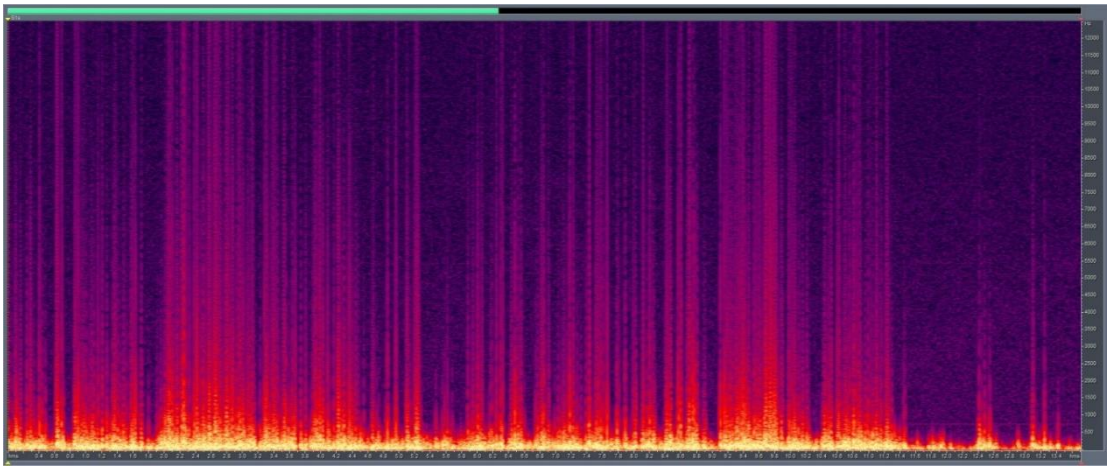

Ilustración 6 – Espectrograma de la clase Saturado

Helicóptero

Esta clase agrupa tanto los eventos de tipo helicóptero (*helix*) como los eventos que recogen vuelos del helicóptero junto a perturbaciones atmosféricas (*wind+helix*).

El helicóptero genera tonos alrededor de los 90 Hz y 170 Hz por encima del fondo (7). Además, el espectrograma del helicóptero se distingue por tener marcas en frecuencias de hasta 4000/5000 Hz. Además, tiene franjas moradas verticales, debidas a los cambios de velocidad. En la imagen se muestra el espectrograma de un fragmento del *sol 133* donde se puede ver este fenómeno.

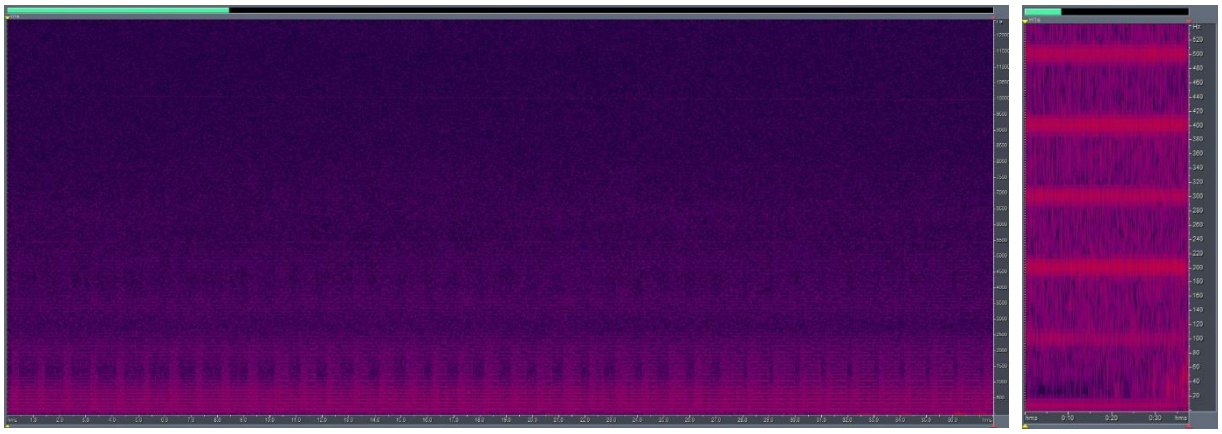

Ilustración 7 – Espectrograma de la clase Helicóptero

Compresor del instrumento MOXIE

Como en el caso anterior, la clase compresor (*moxie*) agrupa tanto los eventos en los que el compresor se puede escuchar sin ruido ambiente como en los que se pueden apreciar pequeñas perturbaciones de fondo (*wind+moxie*). El instrumento *Moxie*

funciona cada 1-2 meses para producir O_2. El objetivo principal de estas operaciones repetidas es buscar la posible degradación de la eficacia de la producción de O_2 asociada al entorno de Marte.

Los armónicos más fuertes están cerca de los 500 Hz, en los que también se excitan varias frecuencias más. Este rango corresponde a las frecuencias de resonancia del instrumento MOXIE, como se observó durante pruebas dinámicas (11). A continuación, se muestra un espectrograma de un fragmento del *sol 81* donde se puede ver como los armónicos se sitúan por debajo de 500 Hz.

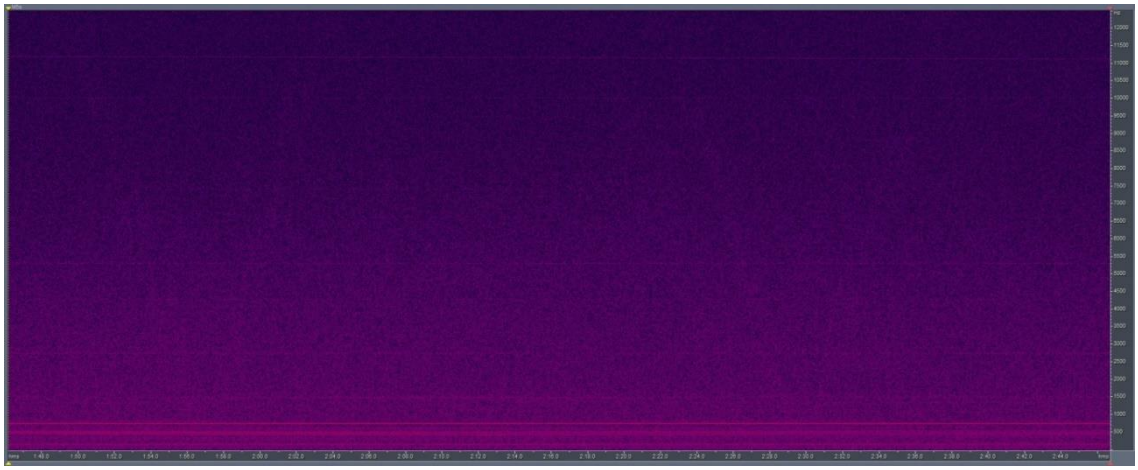

Ilustración 8 – Espectrograma de la clase Compresor

Laser

En uno de los modos del instrumento *SCAM*, el micrófono se utiliza en experimentos *LIBS* (*Laser Induced Break-out Spectroscopy*). Esta técnica utiliza el rayo láser *LIBS* para generar nubes de plasma de las rocas objetivo, analizando las emisiones resultantes para proporcionar información sobre la composición química de la roca (36). Las secuencias típicas de LIBS consisten en 30 disparos de láser en la misma posición realizados a 3 Hz (sólo se registran 29 disparos) (7). Hay dos modos que incluyen la grabación acústica de LIBS. Uno es con una grabación continua durante la ráfaga de láser y el otro es con el micrófono en modo pulsado, en el que se realiza una grabación de 60 ms por cada pulso láser. Este modo de grabación pulsada reduce el volumen de datos mientras de datos, aunque se capta el pulso del láser y los ecos asociados (38).

Este espectrograma cuenta con franjas verticales muy característicos con potencia repartida sobre todo entre 2000 y 14000 Hz que se pueden observar en la siguiente captura tomada durante el *sol 37*.

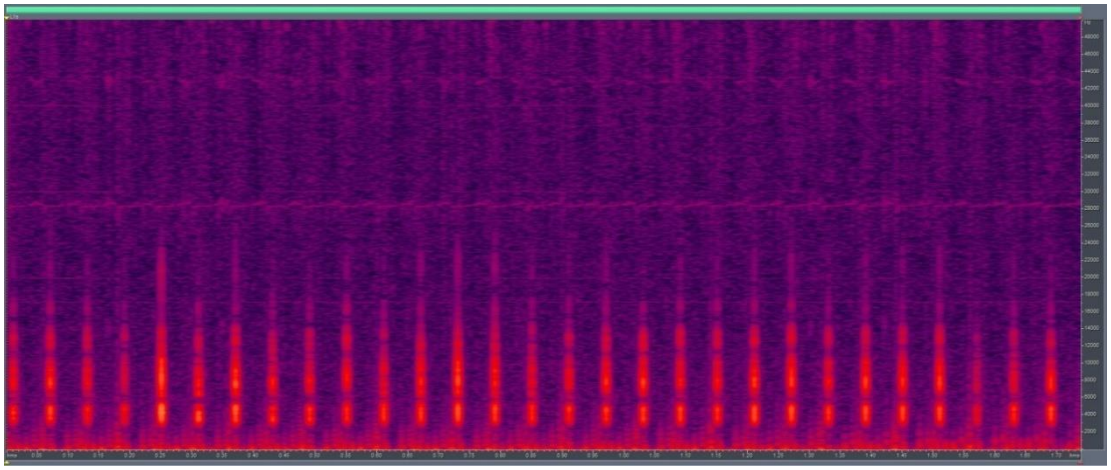

Ilustración 9 – Espectrograma de la clase Laser

No evento (rest)

Las zonas donde no se ha marcado un evento dentro de los ficheros incluidos en el análisis, han sido marcadas como *no event*os. En estos ficheros, se puede escuchar el ruido particular del ambiente marciano. A continuación, se muestra un espectrograma de un fichero del *sol 297* como ejemplo, donde se puede ver que hay un solo componente que se encuentra por encima de 780 Hz. Este armónico está siempre presente y no se ha identificado en la documentación del instrumento por lo que podría ser un ruido asociado al funcionamiento de los sistemas eléctricos del *rover*.

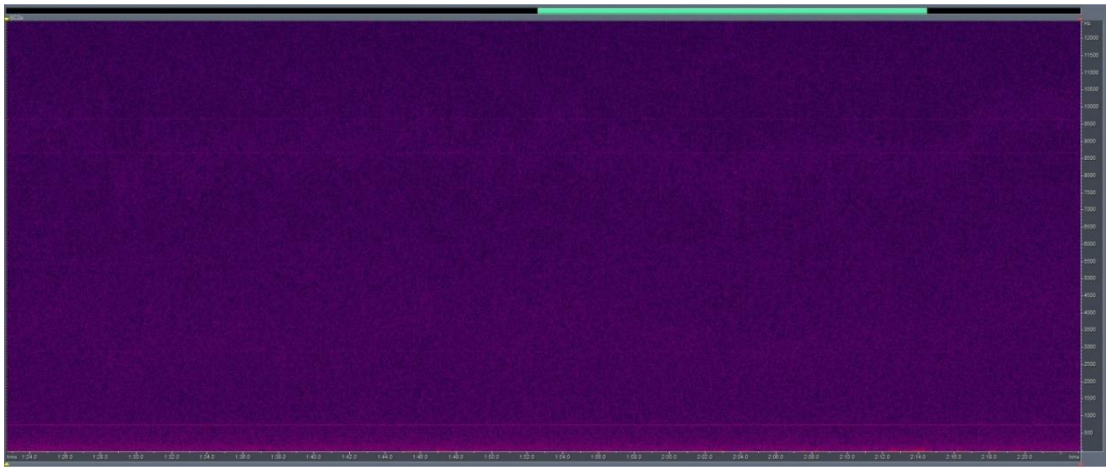

Ilustración 10 – Espectrograma de la clase No evento (*rest*)

Eventos de interés

Varios eventos de instrumentos del *rover* sin identificar se han encontrado entre los eventos de ruido atmosférico. Para ser más concreto, se han encontrado dos ficheros

en la última grabación registrada tras una serie de eventos de la clase *Laser* del *sol* 279 y el *sol* 297 en los que se produce este fenómeno.

En el espectrograma tomado durante el *sol 297* de los eventos se pueden ver líneas verticales con armónicos separados entre sí que concentran su potencia en las frecuencias más bajas. La intensidad del sonido crece según va trascurriendo el tiempo para decaer hasta desparecer por completo. Estos eventos parecen recoger sonidos producidos por experimentos LIBS haciendo variar la potencia del láser y produciendo efectos acústicos de intensidad variable. La *información disponible en los documentos Mission Lead (ML-DL)* y *Documentarian (T*-Doc) del *Perseverance Analyst's Notebook* (39) para el *sol* 279 y 297 recogen más información sobre los objetivos de estos experimentos LIBS.

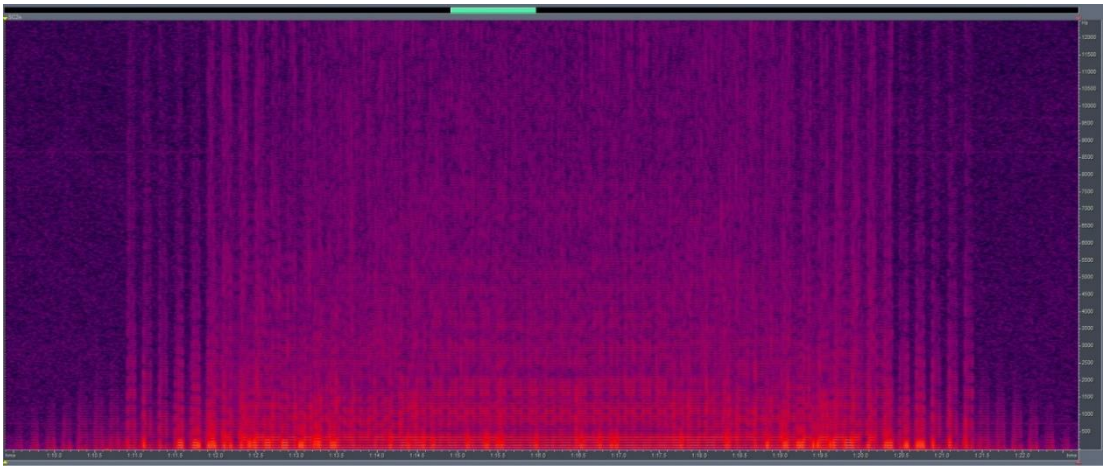

Ilustración 11 – Espectrograma de un evento de interés

4.1.4. ANÁLISIS DE LA DURACIÓN DE LAS MUESTRAS

Como se han etiquetado eventos con diferentes longitudes, se ha realizado un estudio de su duración en segundos obteniendo una media de 5,72 segundos. Además, se ha calculado el histograma y en este se puede ver que la mayor parte de las muestras tienen longitudes de entre 0 y 2 segundos, siendo mayoritarias las muestras de entre 1 y 2 segundos. Por esta razón se ha elegido longitudes de 1 y 2 segundos para realizar los experimentos.

Ahora, se muestran los histogramas con el número de eventos para cada longitud.

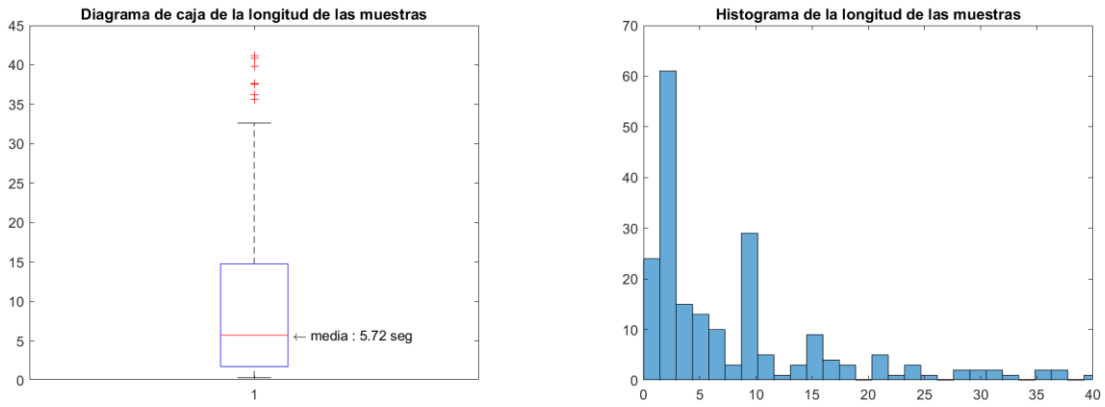

Ilustración 12 – Histograma con la longitud de las muestras

4.2. RED NEURONAL PARA LA CLASIFICACIÓN DE AUDIO

4.2.1. PREPARACIÓN DE DATOS PARA LA RED NEURONAL

Después de analizar la duración de las muestras y dado que algunas clases no cuentan con suficientes muestras como para hacer un análisis con longitud de 2 segundos (por ejemplo, *saturated*), se ha decidido realizar la clasificación con fragmentos de señal de 1 segundo, obteniendo un total de 4343 muestras. En el experimento con longitud de ventana de 2 segundos hay 2060 muestras y hay 7525 muestras cuando se ha configurado la superposición de muestras cada 500 milisegundos en el experimento de *sobremuestreado*.

Para la realización de los cortes en fragmentos de las grabaciones originales, se ha implementado el 'Script prep_mars_db.py', programado en *Python* que genera los ficheros de datos con los nombres de ficheros y clases que servirán de entrada en la ejecución de la red. Su tarea principal es generar unas listas en las que se incluya el principio, el final, el nombre y la clase del evento, *cortar* los ficheros de audio y generar unos ficheros de datos con el nombre del fragmento y la clase (por ejemplo, brisa, moxie, helicóptero, resto...)

A continuación, se muestra un esquema que describe el funcionamiento del script.

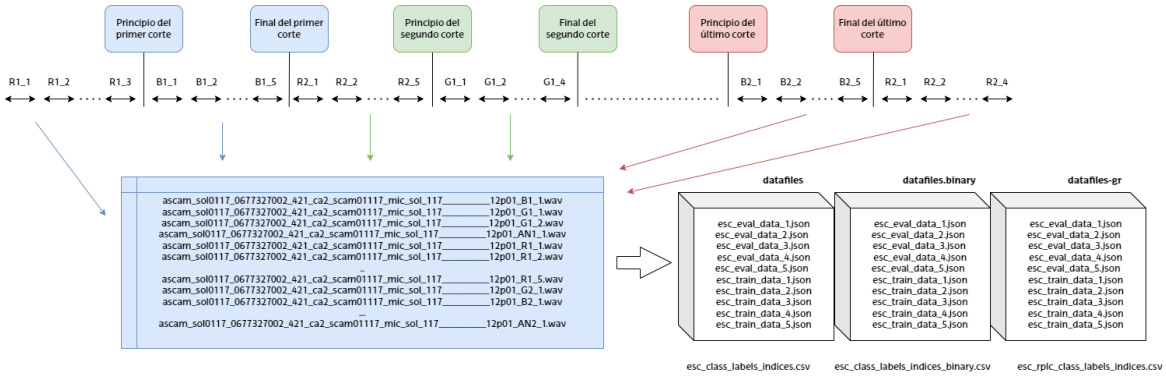

Ilustración 13 – Esquema funcionamiento script de cortado

Adicionalmente, el script permite configurar el número de fragmentos máximos del tipo 'Resto' y agrupar clases en una sola para optimizar los resultados. La descripción de estas variables puede verse en el anexo "Script prep_mars_db.py".

El script permite, además, realizar *sobremuestreado*, cambiando los parámetros definidos en el anexo. Cuando se realiza el *sobremuestreado* se realiza solapamiento temporal cogiendo un porcentaje de intervalo intermedio en los fragmentos. Por ejemplo, sin la opción de solapamiento, las muestras se separan desde el segundo 10 al 11, del 11 al 12, etc. Cuando se pone un valor de solapamiento de 0.5, el script separa las muestras del segundo 10 al 11, del 10.5 al 11.5, del 11 al 12, del 11.5 al 12.5, etc. De esta manera se obtiene un número mayor de muestras para introducir en la red neuronal. Esta tarea, se realiza solamente en las clases etiquetadas y no en las clases de tipo *Resto*, ya que hay muchos más eventos en este grupo, aunque este comportamiento se puede modificar variando un parámetro en la llamada.

El modelo *AST* (12) realiza una validación cruzada para la evaluación del rendimiento del modelo. En la validación cruzada, la base de datos se divide en diferentes carpetas y se realizan varias iteraciones de entrenamiento-evaluación. En cada una de ellas una carpeta diferente es utilizada como subconjunto para la evaluación. Mientras que para el entrenamiento el modelo utiliza el resto de carpetas juntas. Por ejemplo, para la carpeta número 1, el *folder* 1 forma el subconjunto de evaluación (*eval*) y los *folder* 2, 3, 4 y 5 componen el subconjunto de entrenamiento (*train*). Para más información, consultar el anexo 'Estructura del Directorio de Trabajo'.

Por ello, los datos se han divido en cinco carpetas diferentes con un tamaño similar, tal y como se muestra en la siguiente tabla:

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Total
breeze	82	82	82	82	81	409
gust_of_wind	52	52	51	51	51	257
atmospheric_noise	62	62	61	61	61	307
saturated	8	8	8	8	7	39
helix	74	74	74	74	73	369
wind+helix	5	5	4	4	4	22
moxie	200	200	200	200	199	999
wind+moxie	86	85	85	85	85	426
Laser	35	34	34	34	34	171
rest	269	269	269	269	268	1344
Evento	643	638	636	635	632	3184
No evento	295	295	295	295	294	1474
Total	873	871	868	868	863	4343

Tabla 4 – Número de eventos en cada carpeta

Por último, *AST* permite realizar un equilibrado de las muestras consideradas durante el entrenamiento añadiendo un fichero con peso para cada evento en otro fichero de datos mediante una llamada al 'Script gen_weight_file.py'. Estos ficheros serán utilizados por la función *WeightedRandomSampler* de la librería *pytorch* para hacer una ejecución balanceada.

El script genera un fichero de *log* que permite comprobar el número total y el número de eventos por clase en cada carpeta (lo indicado en la Tabla 4). Para más información, consultar el anexo 'Script prep_mars_db.py'.

4.2.2. ADAPTACIÓN DE LA RED AST PARA LA CLASIFICACIÓN DE SONIDOS

Como se ha descrito anteriormente, se ha optado por utilizar el proyecto *Audio Spectrogram Transformer* (AST) (1) como punto de partida para la creación de la red neuronal. Este proyecto se publicó en Abril del 2021 puede descargarse de la siguiente *url* (12):

https://github.com/YuanGongND/ast

El proyecto cuenta con tres *recetas* diferentes; *ESC-50* (34), *Speechcommands V2* (35) y *Audioset* (33). Debido a que se trata de un ejemplo similar, se ha decidió utilizar la receta *ESC-50* como punto de partida para el trabajo. El conjunto de datos *ESC-50* es una colección etiquetada de 2.000 grabaciones de ambiente adecuadas para la evaluación comparativa de los métodos de clasificación de sonido. Este conjunto de datos se compone de grabaciones de 5 segundos de duración organizadas en 50 clases semánticas (con 40 ejemplos por clase) en 5 categorías principales (34).

Esta *receta* necesita como entradas, los ficheros de audio con los cortes que se utilizarán para entrenar la red. Un fichero con la descripción de las clases que se van a utilizar en el proyecto y los ficheros de datos donde se reparten las muestras en las diferentes carpetas de entrenamiento y evaluación.

Para generar todos estos ficheros, se han programado dos scripts cuyo contenido se describe en los anexos 'Script prep_mars_db.py' y 'Script get_input_files.py'. Estos ficheros se encargan de descargar y etiquetar los ficheros de audio (*get_input_files.py*) de la web de la NASA (10) para después *cortarlos*, realizar el *clustering* y crear la estructura de carpetas necesaria para la ejecución de la red (*prep_mars_db.py*), obteniendo una estructura de ficheros descrita en el anexo 'Estructura del Directorio de Trabajo'.

El entrenamiento de la red neuronal requiere el ajuste de algunos parámetros como son por ejemplo *epoch* o tamaño del lote (*batch size*). El número *epoch* es un parámetro que define el número de veces que el algoritmo de aprendizaje pasará a través de todo el conjunto de datos de entrenamiento. El modelo utiliza los resultados del mejor *epoch* para el cálculo de métricas asociadas a la calidad de la clasificación. El *tamaño del lote* identifica el número de muestras de entrenamiento utilizados en una iteración. Este parámetro está limitado por la memoria de la *unidad de procesamiento gráfica* (GPU) donde se ejecuta el proceso y condiciona la velocidad con la que se entrena la red.

5. EXPERIMENTOS Y RESULTADOS

5.1. EVALUACIÓN DE LOS CLASIFICADORES

En este capítulo se describen algunos conceptos sobre la evaluación de los sistemas de clasificación que luego se utilizarán en el capítulo 'Descripción de los Experimentos' como son las matrices de confusión, las diferentes métricas y tipos de puntuación.

5.1.1. MATRIZ DE CONFUSIÓN

Una matriz de confusión es una forma tabular de visualizar el rendimiento de un modelo de predicción. Cada entrada de una matriz de confusión denota el número de predicciones realizadas por el modelo en las que clasificó las clases correcta o incorrectamente (40).

Para simplificar, consideremos que nuestro problema de clasificación multiclase es un problema de clasificación de 3 clases. Digamos que tenemos el conjunto de datos que tiene tres etiquetas, viento, laser y normal. La siguiente es una posible matriz de confusión para estas clases.

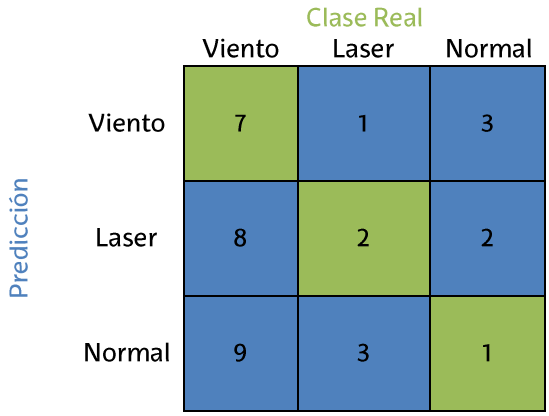

Ilustración 14 – Matriz de confusión para la clasificación multiclase

Los valores de la diagonal principal se corresponden con los valores estimados de forma correcta por el modelo.

5.1.2. MÉTRICAS DE EVALUACIÓN

Para realizar una clasificación binaria se utilizan los siguientes conceptos:

- **Verdadero Positivo (TP):** Número de predicciones en las que el clasificador predice correctamente la clase positiva como positiva.

- **Verdadero negativo (TN):** Número de predicciones en las que el clasificador predice correctamente la clase negativa como negativa.
- **Falso positivo (FP):** Número de predicciones en las que el clasificador predice incorrectamente la clase negativa como positiva.
- **Falso negativo (FN):** Número de predicciones en las que el clasificador predice incorrectamente la clase positiva como negativa.

En cambio, para un clasificador multiclase se utilizan los siguientes conceptos:

- **OvR (One vs Rest):** Como su nombre indica, es un método para evaluar modelos multiclase comparando cada clase con todas las demás al mismo tiempo. En este escenario, tomamos una clase y la consideramos como nuestra clase "positiva", mientras que todas las demás (el resto) se consideran como la clase "negativa". Al hacer esto, reducimos el resultado de la clasificación multiclase a uno de clasificación binaria para cada una de las clases, y así es posible utilizar todas las métricas de clasificación binaria conocidas para evaluar este escenario (41).
- **OvO (One vs One):** En este caso, se compara todas las posibles combinaciones de dos clases del conjunto de datos. En un escenario de 3 clases y elegimos la combinación "Clase1 vs Clase2" como la primera. El primer paso es obtener una copia del conjunto de datos que sólo contenga las dos clases y descartar todas las demás. A continuación, definimos las observaciones con clase real = "Clase1" como nuestra clase positiva y las que tienen clase real = "Clase2" como nuestra clase negativa. Ahora que el problema es binario, también podemos utilizar las mismas métricas que utilizamos para la clasificación binaria. Hay que tener en cuenta que "Clase1 vs. Clase2" es diferente de "Clase2 vs. Clase1", por lo que deben contabilizarse ambos casos. Por ello, en un conjunto de datos de 3 clases obtenemos 6 puntuaciones OvO, y en un conjunto de datos de 4 clases obtenemos 12 puntuaciones OvO (41).

Las estrategias OvR y OvO deben ser utilizadas para adaptar cualquier métrica de clasificación binaria a la tarea de clasificación multiclase. En nuestro caso se calcularán las métricas usando la estrategia OvR.

En ambas estrategias tenemos un conjunto de métricas diferentes para cada clase, por lo que para evaluar el rendimiento del clasificador en su globalidad es necesario hacer un promedio de las mismas. Este promediado puede hacerse según diferentes criterios:

- **Macro:** Calcula las métricas para cada clase individualmente y luego toma la media no ponderada de las medidas.
- **Micro:** puntuación micro-promediada y se calcula considerando el total de TP, el total de FP y el total de FN del modelo. No considera cada clase individualmente, sino que calcula las métricas globalmente. Cuando calculamos las métricas globalmente, todas las medidas se igualan.
- **Ponderada (*weighted*):** El último es el promedio ponderado de la métrica donde el peso de cada clase es el número total de muestras de esa clase.

Se ha decidido utilizar el promedio de tipo *macro* para el cálculo de las métricas ya que el número de eventos en las diferentes clases no es homogéneo.

La precisión de un clasificador multiclase se calcula como la precisión media por tipo. Esto nos da una idea de la eficacia del clasificador a nivel de cada clase. Siempre es mejor utilizar la matriz de confusión como criterio de evaluación para su modelo de aprendizaje automático. Aun así, se pueden utilizar otras medidas de rendimiento más comunes.

- **Exactitud (accuracy):** Le proporciona la exactitud global del modelo, es decir, la fracción del total de muestras que fueron clasificadas correctamente por el clasificador.
- **Tasa de clasificación errónea:** Indica qué fracción de predicciones son incorrectas. También se conoce como error de clasificación.
- **Precisión:** Indica qué fracción de las predicciones como clase positiva fueron realmente positivas.
- **Sensibilidad (*Recall*):** Indica qué fracción de todas las muestras positivas fueron predichas correctamente como positivas por el clasificador. También se conoce como probabilidad de detección.
- **Puntuación F1:** Combina la *precisión* y el *recall* en una sola medida. Matemáticamente es la media armónica de la precisión y el *recall*. Para un clasificador multiclase de *k* clases se calcula como (42):

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

- **Curva ROC (receiver operating characteristic curve):** es un gráfico que muestra el rendimiento de un modelo de clasificación en todos los umbrales de clasificación. Esta curva traza dos parámetros, la Tasa de Verdaderos Positivos (TPR):TP/(TP+FN) frente la tasa de falsos positivos (FPR): FP/(FP+TN). Para utilizar las curvas ROC y el AUC ROC en la clasificación multiclase, se calcula un ROC para cada clase comparando el: OvR y OvO.
- **AUC: Área bajo la curva ROC**: Indica en qué medida el modelo es capaz de distinguir entre las clases. Cuanto mayor sea el AUC, mejor será el modelo para predecir las clases de forma correcta. Para un clasificador multiclase de *k* clases, se calcula el valor para cada clase y se hace una media.

5.2. DESCRIPCIÓN DE LOS EXPERIMENTOS

En este capítulo, se describen los experimentos realizados para después describir los resultados obtenidos y así evaluar la calidad de la *Red Neuronal* utilizada para clasificar los sonidos en Marte.

Se han evaluado 9 Modelos diferentes, completando un total de seis experimentos que se describen a continuación.

Etiqueta	Descripción	Preentrenamiento	Ventana (ms)	Fuente
M1	Pequeñas perturbaciones con *breeze*.	Si	1000	*fits*
M2	Sin modelos preentrenados	No	1000	*fits*
M3	Eventos recogidos de ficheros *wav*	Si	1000	*wav*
M4	Binario (Detección de Eventos)	Si	1000	*fits*
M5	Longitud de ventana 2 segundos	Si	2000	*fits*
M6	Pequeñas perturbaciones con *Rest*.	Si	1000	*fits*
M7	Solapamiento 0.5	Si	1000	*fits*
M8	Equilibrado	Si	1000	*fits*
M9	Equilibrado + Solapamiento	Si	1000	*fits*

Tabla 5 – Definición de los diferentes modelos generados

Se ha elegido el mejor resultado entre 25 *epoch* para obtener las métricas en todos casos. Se puede concluir que la red neuronal se comporta de forma correcta en todos los *epoch* con resultados cercanos al 89%. En cuanto al *tamaño de lote*, se ha elegido un valor de 12, con el que se llenan más de 11GB durante la ejecución.

5.2.1. EXPERIMENTO 1: PREENTRENAMIENTO

En este experimento se evalúa si la precisión aumenta cuando la red realiza el aprendizaje con pre-entrenamiento. Además, se han utilizado los ficheros *fits* como origen de la información. Como se ha indicado anteriormente, en el modelo M2 están

los datos obtenidos sin *preentrenamiento*. Ambos modelos han sido entrenados con la base de datos de fragmentos de 1 s sin solapamiento que tiene el siguiente número de muestras:

	Folder	laser	breeze	helix	moxie	saturated	event	rest	no event	Total
M1 M2 M3 M4 M8	fold1	35	195	79	285	8	602	269	269	871
	fold2	34	195	78	285	8	600	269	269	869
	fold3	34	195	78	285	8	600	269	269	869
	fold4	34	194	78	285	8	599	269	269	868
	fold5	34	194	78	285	7	598	268	268	866
	Total	**171**	**973**	**391**	**1425**	**39**	**2999**	**1344**	**1344**	**4343**

Tabla 6 – Número de muestras en para cada folder en los modelos 1,2, 3, 4 y 8

A continuación, se muestran los resultados:

	M1				
	ACC	AUC	PREC	RECALL	F1
F1	89.08%	97.82%	88.92%	90.21%	87.26%
F2	91.42%	97.25%	91.72%	87.53%	89.26%
F3	89.48%	98.70%	96.42%	91.00%	92.84%
F4	90.66%	99.09%	97.37%	94.87%	94.98%
F5	95.91%	99.29%	89.59%	87.88%	87.64%
T	**91.31%**	**98.43%**	**92.80%**	**90.30%**	**90.40%**

	M2				
	ACC	AUC	PREC	RECALL	F1
F1	81.97%	96.31%	82.46%	80.21%	81.77%
F2	90.91%	98.16%	90.90%	86.98%	89.84%
F3	85.85%	97.76%	81.95%	82.10%	82.03%
F4	83.64%	98.23%	84.18%	84.89%	84.59%
F5	93.42%	98.78%	89.60%	84.18%	85.46%
T	**87.16%**	**97.85%**	**85.82%**	**83.67%**	**84.74%**

Tabla 7 – Resultados de las métricas para cado modelo en el experimento 1

De la evaluación de las métricas de este experimento se concluye que se obtiene en torno a un 3% de mejora utilizando el modelo *pre-entrenado* y que la diferencia de resultados se ve reflejada principalmente en el primer *folder*.

A continuación, se muestra las matrices de confusión:

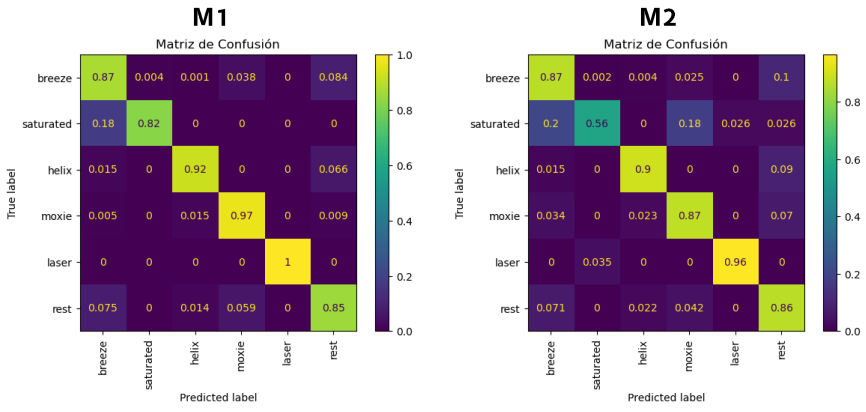

Ilustración 15 – Matrices de confusión para el experimento 1

En los resultados del modelo no preentrenado (M2) se puede ver como se produce perdida en la *exactitud* en la clasificación de todas las clases al no activar esta característica.

5.2.2. EXPERIMENTO 2: DATOS NORMALIZADOS

Tal y como se explica en capítulo 'Base de Datos de Eventos Sonoros'. Los ficheros con los eventos de sonido se encuentran en formato *fits* y *wav*. En el caso de los ficheros *fits*, la información se encuentra sin normalizar. En este capítulo se compara la precisión obtenida cuando se utilizan los datos previamente procesados M3. En este caso también ambos modelos cuentan con el mismo número de muestras, por lo tanto, están repartidas de la misma manera que en la Tabla 6.

	M1				
	ACC	AUC	PREC	RECALL	F1
F1	90.01%	97.77%	92.87%	91.95%	91.30%
F2	89.30%	98.39%	96.44%	91.72%	89.83%
F3	87.57%	98.37%	94.96%	89.54%	91.22%
F4	91.01%	99.22%	97.94%	94.86%	93.21%
F5	95.61%	99.40%	93.86%	83.37%	84.12%
T	**90.70%**	**98.63%**	**95.22%**	**90.29%**	**89.94%**

	M3				
	ACC	AUC	PREC	RECALL	F1
F1	85.65%	97.43%	88.91%	89.13%	81.75%
F2	89.07%	97.57%	94.12%	92.69%	90.65%
F3	77.79%	97.95%	93.22%	84.54%	85.21%
F4	89.17%	99.13%	97.65%	93.98%	93.20%
F5	96.54%	99.62%	94.53%	86.33%	88.04%
T	**87.64%**	**98.34%**	**93.69%**	**89.34%**	**87.77%**

Tabla 8 – Resultados de las métricas para cado modelo en el experimento 2

Comparando los modelos M1 y M3 se puede comprobar que se obtiene una precisión 3% mayor cuando se utiliza como fuente de información la recogida en la tabla *sound* de los ficheros *fits*.

En las matrices de confusión se puede que a pesar de que los resultados sean bastante similares, se obtienen una mejora significativa en la clasificación que hace que la precisión sea mejor cuando se utilizan los ficheros *fits*.

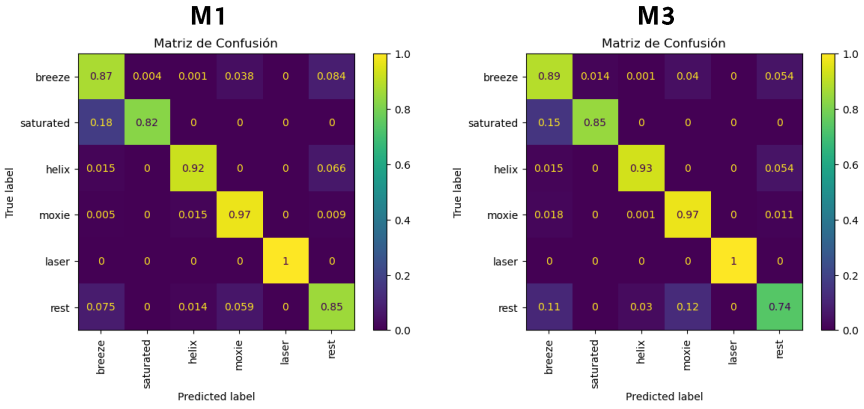

Ilustración 16 – Matrices de confusión para el experimento 2

Para ser más concreto, se pierde precisión en las clases *rest*, *helix* y *moxie*. al utilizar los ficheros de audio normalizados.

5.2.3. EXPERIMENTO 3: DETECCIÓN DE EVENTOS

Mediante este experimento se obtiene la precisión a la hora de diferenciar un fenómeno sonoro de una situación normal. Como se ve en los resultados de la siguiente

tabla, la precisión es un 2% mayor para la detección binaria (Modelo M4) y aunque este resultado es positivo, nos indica que la red está equivocándose a la hora de clasificar alguno de los fragmentos marcados como suceso. El modelo M4 está repartido según el número de muestras (*event, no event*) mostrado en la Tabla 6.

	M1				
	ACC	AUC	PREC	RECALL	F1
F1	90.01%	97.77%	92.87%	91.95%	91.30%
F2	89.30%	98.39%	96.44%	91.72%	89.83%
F3	87.57%	98.37%	94.96%	89.54%	91.22%
F4	91.01%	99.22%	97.94%	94.86%	93.21%
F5	95.61%	99.40%	93.86%	83.37%	84.12%
T	**90.70%**	**98.63%**	**95.22%**	**90.29%**	**89.94%**

	M4				
	ACC	AUC	PREC	RECALL	F1
F1	88.75%	92.19%	92.26%	87.75%	87.06%
F2	94.48%	97.88%	97.58%	93.33%	93.51%
F3	90.22%	95.23%	93.61%	89.94%	88.85%
F4	93.32%	97.37%	96.66%	93.11%	92.33%
F5	98.04%	99.40%	99.11%	97.65%	97.70%
T	**92.96%**	**96.41%**	**95.84%**	**92.36%**	**91.89%**

Tabla 9 – Resultados de las métricas para cado modelo en el experimento 3

De la matriz obtenida en este experimento se concluye que la definición de los eventos tipo *resto* puede complicar la clasificación de eventos en el proyecto. Algunos eventos como el *ruido atmosférico* o el *helicóptero* se confunden con el ruido de fondo de la atmósfera debido a la poca intensidad de los eventos y a la fuerte atenuación de la transmisión de altas frecuencias en la atmósfera marciana (7).

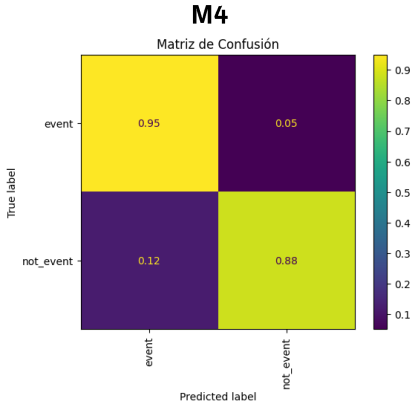

Ilustración 17 – Matriz de confusión para el experimento 3

5.2.4. EXPERIMENTO 4: EVALUACIÓN DEL TAMAÑO ÓPTIMO DE VENTANA

En el cuarto experimento, se evalúan los resultados cuando se varía el tamaño de la ventana entre 1 y 2 segundos. El modelo M5 tiene un tamaño de ventana de 2 segundos mientras que el M1 se ha entrenado con la base de datos con el número de muestras indicado en la en la Tabla 6 (ventana de 1 segundo). Para el caso del modelo M5, el número de muestras de 2 segundos es el siguiente:

	Folder	laser	breeze	helix	moxie	saturated	event	rest	no event	Total
M5	fold1	13	94	38	139	4	288	127	127	415
	fold2	13	93	38	139	4	287	127	127	414
	fold3	13	93	37	139	3	285	127	127	412
	fold4	12	93	37	139	3	284	126	126	410
	fold5	12	93	37	138	3	283	126	126	409
	Total	**63**	**466**	**187**	**694**	**17**	**1427**	**633**	**633**	**2060**

Tabla 10 – Número de muestras en para cada folder en el modelo 5

Los resultados se muestran en la siguiente tabla:

	M1				
	ACC	AUC	PREC	RECALL	F1
F1	90.01%	97.77%	92.87%	91.95%	91.30%
F2	89.30%	98.39%	96.44%	91.72%	89.83%
F3	87.57%	98.37%	94.96%	89.54%	91.22%
F4	91.01%	99.22%	97.94%	94.86%	93.21%
F5	95.61%	99.40%	93.86%	83.37%	84.12%
T	**90.70%**	**98.63%**	**95.22%**	**90.29%**	**89.94%**

	M5				
	ACC	AUC	PREC	RECALL	F1
F1	90.84%	97.79%	92.26%	91.13%	90.64%
F2	91.79%	99.21%	98.12%	93.68%	92.20%
F3	94.17%	99.13%	97.44%	95.31%	95.88%
F4	91.95%	99.26%	98.12%	95.22%	94.08%
F5	95.84%	99.18%	91.35%	80.97%	80.71%
T	**92.92%**	**98.91%**	**95.46%**	**91.26%**	**90.70%**

Tabla 11 – Resultados de las métricas para cado modelo en el experimento 4

De la comparación de estos modelos se puede concluir que con muestras de 2 segundos la clasificación es 2% más precisa. Este resultado, confirma que una ventana mayor hace más fácil la clasificación siempre que la mayor parte de las muestras puedan tener esta duración.

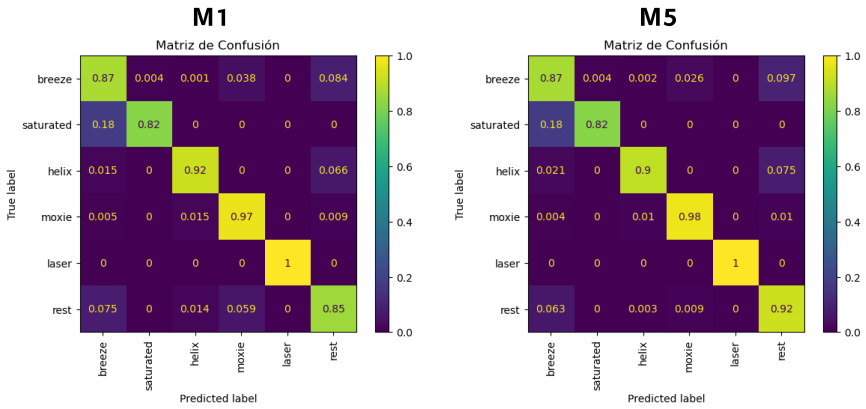

Ilustración 18 – Matrices de confusión para el experimento 4

La matriz del modelo con 2 segundos de ventana indica que se consigue más precisión en todos los tipos de eventos menos *helix*.

5.2.5. EXPERIMENTO 5: EVENTO RUIDO ATMOSFÉRICO

La clasificación inicial para el entrenamiento de la red neuronal de las pequeñas perturbaciones atmosféricas puede ser bastante complicada en algunos casos. Durante la clasificación, se han separado estos fragmentos dentro de la clase *Ruido atmosférico*. Con el *script* desarrollado puede variar el agrupamiento de clases de forma sencilla permitiendo enviar todos estos eventos a la clase *rest* (modelo M6). A continuación, se

evalúan los resultados obtenidos cuando se agrupan estos eventos con la clase *no evento*, como si no hubieran sido etiquetados. El modelo M1 se ha entrenado con la lista de fragmentos estándar según el número de muestras indicado en la en la Tabla 6, mientras que la lista de eventos resultante para entrenar el M6 cuenta con el siguiente número de fragmentos:

	Folder	laser	breeze	helix	moxie	saturated	event	rest	no event	Total
M6	fold1	35	134	79	285	8	541	331	331	872
	fold2	34	133	78	285	8	538	330	330	868
	fold3	34	133	78	285	8	538	330	330	868
	fold4	34	133	78	285	8	538	330	330	868
	fold5	34	133	78	285	7	537	330	330	867
	Total	**171**	**666**	**391**	**1425**	**39**	**2692**	**1651**	**1651**	**4343**

Tabla 12 – Número de muestras en para cada folder en el modelo 6

Los resultados se muestran a continuación:

	M1				
	ACC	AUC	PREC	RECALL	F1
F1	90.01%	97.77%	92.87%	91.95%	91.30%
F2	89.30%	98.39%	96.44%	91.72%	89.83%
F3	87.57%	98.37%	94.96%	89.54%	91.22%
F4	91.01%	99.22%	97.94%	94.86%	93.21%
F5	95.61%	99.40%	93.86%	83.37%	84.12%
T	**90.70%**	**98.63%**	**95.22%**	**90.29%**	**89.94%**

	M6				
	ACC	AUC	PREC	RECALL	F1
F1	85.44%	97.19%	91.43%	88.18%	88.81%
F2	90.67%	97.57%	93.12%	91.50%	91.03%
F3	86.87%	97.86%	93.18%	86.51%	89.14%
F4	87.33%	98.62%	95.12%	91.46%	88.79%
F5	88.12%	98.66%	88.81%	73.56%	75.55%
T	**87.68%**	**97.98%**	**92.33%**	**86.24%**	**86.66%**

Tabla 13 – Resultados de las métricas para cado modelo en el experimento 5

Comparando estos modelos se ha decidido que es mejor agrupar los sucesos con pequeñas perturbación atmosféricas con el resto de fragmentos de viento (Brisa y viento fuerte).

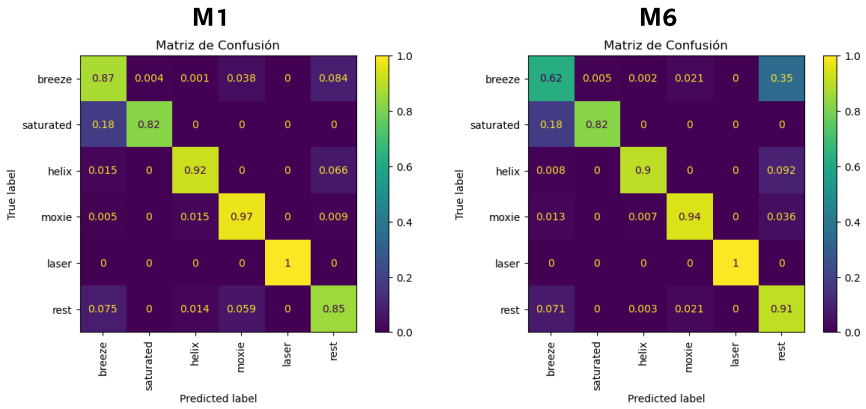

Ilustración 19 – Matrices de confusión para el experimento 5

Al comparar los resultados para estos modelos, se puede ver que la definición de los eventos de ruido atmosférico leve es bastante difícil y es la principal causa de los

errores en la detección de eventos. Clasificar pequeñas perturbaciones puede complicar la tarea de generar una base de datos consistente. Aun así, los resultados globales superiores al 90%, ratifican la consistencia de los procedimientos utilizados.

5.2.6. EXPERIMENTO 6: SOBREMUESTREADO Y EQUILIBRADO

Como se ha indicado en el capítulo 4.2.1, se ha programado un script que permite el solapamiento de las muestras para obtener un número mayor de muestras, llegando a un total de 7525 muestras. El solapamiento no se realiza en la clase del tipo resto por ser esta mayoritaria y de esta manera se obtiene un resultado más equlibrado que supone una mejora de un 1% en la precisión. Además, el modelo AST, permite obtener resultados *balanceandos*, añadiendo pesos a cada muestra en cuada uno de los ficheros de datos. Tanto para el caso normal y del entrenamiento balanceado el número de muestras se ajusta a lo descrito en la Tabla 6, en el caso del sobremuestreado y sobremuestreado y balanceado el número de muestras es mayor y se ajusta a lo que se describe en la siguiente tabla:

	Folder	laser	breeze	helix	moxie	saturated	event	rest	no event	Total
M7 M9	fold1	73	409	160	579	17	1238	269	269	1507
	fold2	72	409	160	579	17	1237	269	269	1506
	fold3	72	409	159	579	17	1236	269	269	1505
	fold4	72	409	159	579	17	1236	269	269	1505
	fold5	72	409	159	578	16	1234	268	268	1502
	Total	**361**	**2045**	**797**	**2894**	**84**	**6181**	**1344**	**1344**	**7525**

Tabla 14 – Número de muestras para cada folder en los modelos 7 y 9

En este experimento, se comparan los resultados de la ejecución sin *sobremuestreado* ni balanceado (M1), frente al modelo con *sobremuestreado* (M7), el modelo con balanceado (M8) y el modelo con *sobremuestreado* y *balanceado* (M9).

	M1				
	ACC	ACC	ACC	ACC	ACC
F1	90.01%	97.77%	92.87%	91.95%	91.30%
F2	89.30%	98.39%	96.44%	91.72%	89.83%
F3	87.57%	98.37%	94.96%	89.54%	91.22%
F4	91.01%	99.22%	97.94%	94.86%	93.21%
F5	95.61%	99.40%	93.86%	83.37%	84.12%
T	**90.70%**	**98.63%**	**95.22%**	**90.29%**	**89.94%**

	M7				
	ACC	ACC	ACC	ACC	ACC
F1	91.57%	98.03%	90.71%	92.16%	91.23%
F2	87.52%	97.03%	93.79%	90.20%	90.73%
F3	90.76%	98.61%	95.13%	90.18%	91.11%
F4	92.09%	99.32%	97.33%	94.81%	92.72%
F5	96.80%	99.49%	92.13%	90.85%	92.28%
T	**91.75%**	**98.50%**	**93.82%**	**91.64%**	**91.61%**

	M8				
	ACC	ACC	ACC	ACC	ACC
F1	89.67%	97.84%	92.32%	91.52%	88.28%
F2	86.19%	97.43%	95.39%	91.24%	86.15%
F3	92.52%	98.83%	96.51%	94.60%	94.22%
F4	83.76%	98.60%	95.10%	91.01%	88.52%
F5	95.84%	99.11%	91.81%	85.86%	87.71%
T	**89.60%**	**98.36%**	**94.23%**	**90.85%**	**88.98%**

	M9				
	ACC	ACC	ACC	ACC	ACC
F1	89.19%	98.17%	92.33%	90.91%	87.71%
F2	91.73%	99.23%	96.21%	91.16%	89.48%
F3	90.11%	99.01%	95.43%	92.80%	92.65%
F4	88.74%	99.18%	97.38%	94.57%	92.86%
F5	95.14%	99.28%	89.54%	93.30%	91.14%
T	**90.98%**	**98.97%**	**94.18%**	**92.55%**	**90.77%**

Tabla 15 – Resultados de las métricas para cado modelo en el experimento 6

Comparando los resultados de estos modelos se puede concluir que el *sobremuestreado* aporta un 1% de mejora mientras que el *equilibrado* no aporta nunca mejora en los resultados.

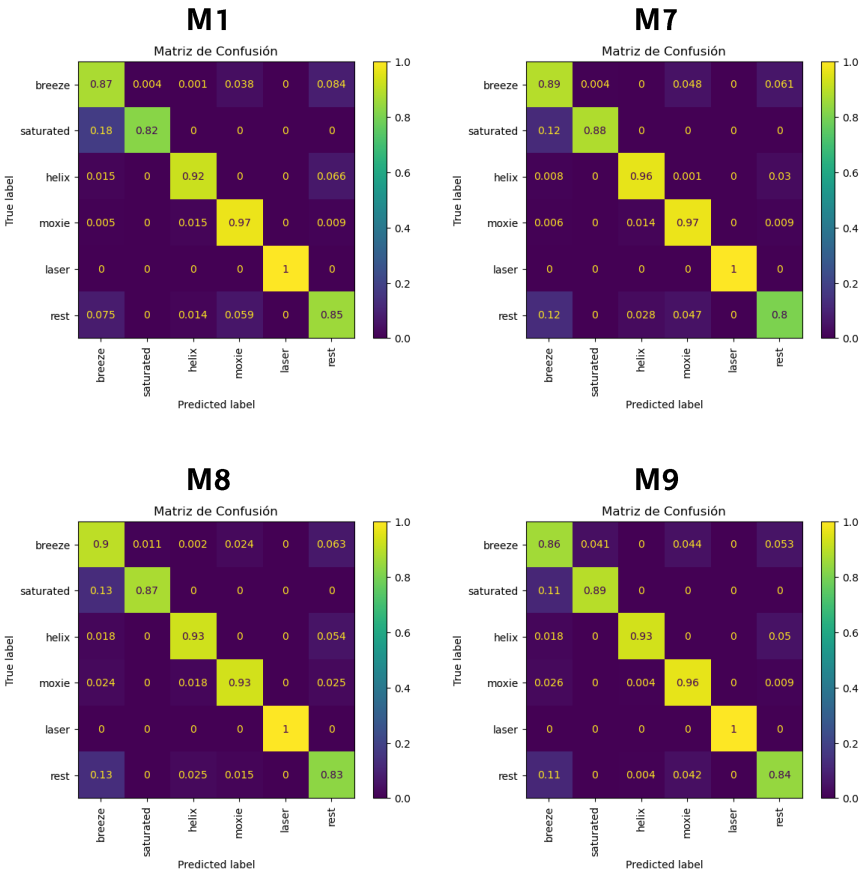

Ilustración 20 – Matrices de confusión para el experimento 6

La matriz de confusión del modelo M7 indica que se consiguen mejoras en todas las clases menos en el resto con *sobremuestreado*.

El modelo M8 no consigue mejorar los resultados de los modelos con solo *sobremuestrado* o *balanceado* y *sobremuestreado*, por lo que se pueden descartar la técnica de *balanceado*.

5.2.7. OTROS DATOS OBTENIDOS

La matriz de confusión del modelo M1 sin normalizar indica que los errores se producen al distinguir los eventos de tipo *resto* y viento. Además, se produce algo confusión al distinguir entre eventos *helix* y *moxie*.

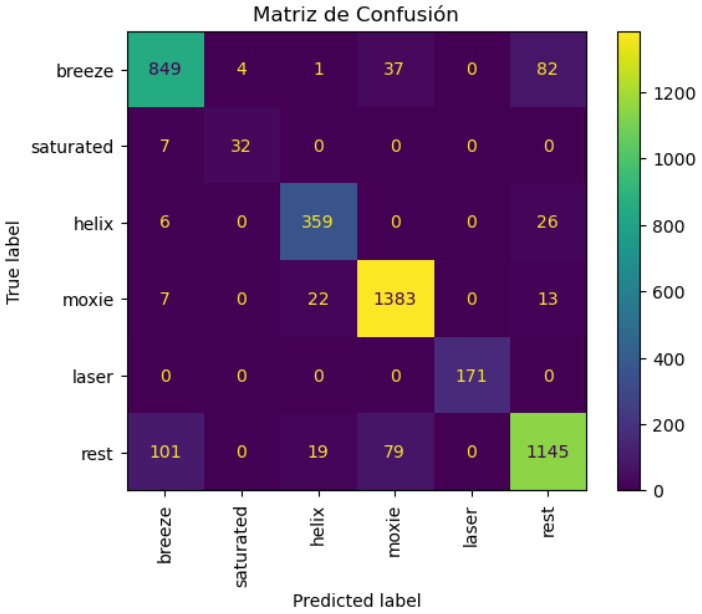

Ilustración 21 – Matriz de confusión para el modelo 1

6. CONCLUSIONES

El desarrollo de este proyecto evalúa la viabilidad del uso de las redes neuronales en la clasificación de sucesos de audio recogidos en Marte. Y a pesar de que el sonido en Marte se transmite de forma diferente a la Tierra (7) (en torno a 100 metros/segundo más lento) y de forma diferente con diferentes velocidades para sonidos graves y agudos, se puede concluir que el uso del *transformer* AST es adecuado para la clasificación de estos eventos, llegando a precisiones superiores al 90%.

La clasificación de las perturbaciones atmosféricas es complicada debida a dos razones. En primer lugar, las perturbaciones no son constantes a lo largo del tiempo y en segundo, se producen pequeñas perturbaciones de muy corta duración que requieren una clasificación muy detallada para obtener precisiones muy altas. Analizando los resultados obtenidos de los sucesos de ruido atmosférico sin y con agrupación, se puede ver que hay una gran confusión cuando todos los fragmentos de viento no se clasifican en una sola clase.

Se obtienen precisiones mayores al agrupar todos los diferentes fragmentos de tipo viento en una sola clase (brisa). Aun así, se ha realizado una clasificación diferenciada que permite llegar a distinguir la intensidad de estos eventos con precisiones globales altas.

Se han encontrados sucesos de sonido del helicóptero y del compresor unidos a eventos de viento. Finalmente, se ha decidido agruparlos para conseguir precisiones más altas.

El uso de modelos *pre-entrenados* (utilizando *ImageNet* y *AudioSet* de *AST*) supone una mejora del 3% respecto a los modelos sin *pre-entrenamiento* y proporciona una clasificación más precisa. Esta transferencia de aprendizaje desde otras aplicaciones es especialmente útil en casos como el nuestro en el que no hay demasiado material de entrenamiento disponible.

A pesar de que etiquetar todos los eventos de los ficheros de audio analizados pueda ser una tarea tediosa, se han conseguido precisiones suficientemente altas que permitan analizar los nuevos ficheros de forma completa.

El *equilibrado* no aporta mejoras en la precisión, por lo que se descarta su uso en futuro. Aun así, se han conseguido mejoras aplicando *solapamiento* en las clases minoritarias

(todas menos el *resto*), por conseguir de esta manera un conjunto más equilibrado. Se puede añadir que se han conseguido precisiones muy altas al igualar el tamaño en muestras de todas las clases duplicando los fragmentos recogidos. Este tipo de *balanceado*, que no se ha documentado en los experimentos, se ha implementado y puede ser útil en el futuro, siempre y cuando se consiga un número similar de muestras en todas las clases que lo permita.

La definición del tamaño en segundos de las muestras no ha quedado bien establecida. Aunque con un tamaño de dos segundos se consigue una precisión mayor, convendría añadir más eventos de tipo *saturado*, *laser* y helicóptero para evaluar este parámetro. Esto implica el etiquetado de más ficheros completos.

Conviene añadir nuevos ficheros con sucesos al proceso para equilibrar el tamaño de las clases. Como se ha indicado anteriormente, los ficheros a incluir deben ser de las clases *saturado*, *laser* y helicóptero.

La ejecución de la red con 25 *epoch* supone días de espera para la obtención de resultados y una gran carga para la GPU. Por esta razón, es conveniente ajustar a un valor menor este parámetro.

El uso de un *Sistemas de Inteligencia Artificial* como el de este trabajo para la clasificación de sonido tiene muchas ventajas. En primer lugar, tiene una alta precisión en la clasificación y permite añadir nuevas clases al sistema solamente realizando la clasificación y etiquetado. Además, la creación de bases de datos libres (*dataset*) para el *machine listening* es imprescindible ya que permite contrastar los resultados obtenidos con otros conjuntos de datos. Desde el punto de vista científico, las tormentas de viento son un fenómeno meteorológico en Marte (43) y los fenómenos acústicos permiten establecer aspectos como el ciclo diario de sonidos en Marte producidos por la actividad atmosférica. Por último, algunos sonidos pueden ayudar a diagnosticar el estado de los equipos del *rover* para realizar tareas de mantenimiento.

7. BIBLIOGRAFÍA

1. **Gong, Y., Chung, Y.-A., Glass, J.** *AST: Audio Spectrogram Transformer. Proc.* s.l. : Proc. Interspeech 2021, 571-575., 2021.

2. **NASA.** Mars 2020 Mission Overview. [En línea] https://mars.nasa.gov/mars2020/mission/overview/.

3. **NASA.** Mars 2020 Mission Contributions to NASA's Mars Exploration Program Science Goal. [En línea] https://mars.nasa.gov/mars2020/mission/science/goals/.

4. **NASA.** Perseverance Rover's Landing Site: Jezero Crater. [En línea] https://mars.nasa.gov/mars2020/mission/science/landing-site/.

5. **NASA.** Microphones on the Perseverance Rover. [En línea] https://mars.nasa.gov/mars2020/spacecraft/rover/microphones/.

6. **NASA.** NASA's Mars Perseverance Rover Provides Front-Row Seat to Landing, First Audio Recording of Red Planet. [En línea] 22 de Febrero de 2021. https://www.nasa.gov/press-release/nasa-s-mars-perseverance-rover-provides-front-row-seat-to-landing-first-audio/.

7. **Maurice, S., Chide, B., Murdoch, N. et al.** *In situ recording of Mars soundscape.* . s.l. : Nature 605, 653–658, 2022.

8. **Petculescu, A. & Lueptow, R. M.** *Atmospheric acoustics of Titan, Mars, Venus, and Earth.* s.l. : Icarus 186, 413–419 (2007)., 2007.

9. **Bass, H. E. & Chambers, J. P.** *Absorption of sound in the Martian atmosphere.* s.l. : J. Acoust. Soc. Am. 109, 3069–3071, 2001.

10. **NASA.** PDS Geosciences Node. [En línea] https://pds-geosciences.wustl.edu/m2020/urn-nasa-pds-mars2020_supercam/data_calibrated_audio/.

11. **NASA.** MOXIE. [En línea] https://mars.nasa.gov/mars2020/spacecraft/instruments/moxie/.

12. **YuanGongND.** AST: Audio Spectrogram Transformer. [En línea] https://github.com/YuanGongND/ast.

13. **Gong, Y., Khurana, S., Rouditchenko, A., & Glass, J.** *CMKD: CNN/Transformer-Based Cross-Model Knowledge Distillation for Audio Classification.* (2022).

14. **Malmgren, Helge.** *Artificial Neural Networks in Medicine and Biology.* Göteborg : s.n., 2000.

15. **Taylor, L.J. Landau and J.G.** *Concepts for Neural Networks.* 1998.

16. **I. Goodfellow, Y. Bengio, and A. Courville.** *Deep Learning*. s.l. : MIT Press, 2016.

17. **F. Eyben, F. Weninger, F. Gross, and B. Schuller.** *Recent developments in openSMILE, the Munich open-source multimedia feature extractor in Multimedia*. 2013.

18. **B. Schuller, S. Steidl, A. Batliner, A. Vinciarelli, K. Scherer, F. Ringeval, M. Chetouani, F. Weninger, F. Eyben, E. Marchi, M. Mortillaro, H. Salamin, A. Polychroniou, F. Valente, and S. K. Kim.** *The Interspeech 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism,*. s.l. : Interspeech, 2013.

19. **Hinton, N. Jaitly and G.** *Learning a better representation of speech soundwaves using restricted boltzmann machines*. s.l. : ICASSP, 2011.

20. **Schrauwen, S. Dieleman and B.** *End-to-end learning for music audio*. s.l. : ICASSP, 2014.

21. **G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou.** *Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network*. s.l. : ICASSP, 2016.

22. **Bengio, Y. LeCun and Y.** *Convolutional networks for images, speech, and time series*. s.l. : The Handbook of Brain Theory and Neural Networks, vol. 3361, no. 10, 1995.

23. **Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley.** *PANNs: Large-scale pretrained audio neural networks for audio pattern recognition*. s.l. : IEEE/ACM TASLP, vol. 28, pp. 2880–2894, 2020.

24. **Y. Gong, Y.-A. Chung, and J. Glass.** *PSLA: Improving audio event classification with pretraining, sampling, labeling, and aggregation*. s.l. : arXiv preprint, 2021.

25. **O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo.** *Streaming keyword spotting on mobile devices*. s.l. : Interspeech, 2020.

26. **P. Li, Y. Song, I. V. McLoughlin, W. Guo, and L.-R. Dai.** *An attention pooling based representation learning method for speech emotion recognition*. s.l. : Interspeech, 2018.

27. **A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby.** *An image is worth 16x16 words: Transformers for image recognition at scale in ICL*. 2021.

28. **H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. J´egou.** *Training data-efficient image transformers & distillation through attention*. s.l. : arXiv preprint arXiv:2012.12877, 2020.

29. **L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, F. E. Tay, J. Feng, and S. Yan.** *Tokens-to-token ViT: Training vision transformers from scratch on ImageNet*. s.l. : arXiv preprint arXiv:2101.11986, 2021.

30. **A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin.** *Attention is all you need*. s.l. : NIPS, 2017.

31. **Ekman, Magnus.** *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, NLP, and Transformers using TensorFlow*. s.l. : Addison-Wesley, 2022.

32. **J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei.** *ImageNet: A large-scale hierarchical image database*. s.l. : CVPR, 2009.

33. **J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen,W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter.** *Audio Set: An ontology and human-labeled dataset for audio events*. s.l. : ICASSP, 2017.

34. **Piczak, K. J.** *ESC: Dataset for environmental sound classification*. s.l. : Multimedia, 2015.

35. **Warden, P.** *Speech commands: A dataset for limited-vocabulary speech recognition*. s.l. : arXiv preprint arXiv:1804.03209, 2018.

36. **Dorothea DeLapp, Paolo Pilleri.** *Supercam PDS User Guide*. 2022.

37. **Geosciences Node Planetary Data System.** Perseverance (Mars 2020) Analyst's Notebook. [En línea] https://an.rsl.wustl.edu/m20/AN/account/login.aspx.

38. **Sylvestre Maurice, Roger C. Wiens.** *The SuperCam Instrument Suite on the NASA Mars 2020 Rover: Body Unit and Combined System Tests*. Los Alamos : Springer, 2020. https://doi.org/10.1007/s11214-020-00777-5.

39. **NASA Planetary Data System Geosciences.** Perseverance (Mars 2020) Analyst's Notebook. [En línea] https://an.rsl.wustl.edu/m20/AN/an3.aspx.

40. **Mohajon, Joydwip.** Confusion Matrix for Your Multi-Class Machine Learning Model. [En línea] 29 de May de 2020. https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826.

41. **Vinícius Trevisan.** Multiclass classification evaluation with ROC Curves and ROC AUC. [En línea] 12 de Febrero de 2022. https://towardsdatascience.com/multiclass-classification-evaluation-with-roc-curves-and-roc-auc-294fd4617e3a.

42. **Foss, Vicki.** Multiclass Classification Model Evaluation. [En línea] 24 de August de 2019. https://parasite.id/blog/2018-12-13-model-evaluation/.

43. **NASA.** Mars Report: Dust Storms on Mar. [En línea] 14 de Febrero de 2022. https://mars.nasa.gov/resources/26555/mars-report-dust-storms-on-mars/.

44. **Anaconda, Inc.** CONDA. [En línea] 2017. https://docs.conda.io/en/latest/.

45. **Josephernest.** wave.py (enhanced). [En línea] 28 de Febrero de 2019. https://gist.github.com/josephernest/e3903ba30b820cd199500e50f145a11f.

8. ANEXOS

8.1. ESTRUCTURA DEL DIRECTORIO DE TRABAJO

La estructura del directorio de trabajo de la red neuronal varía en función de la fase de ejecución en el que nos encontremos. La ejecución del proyecto tiene tres fases:

1. Fase inicial: No existen cortes con eventos, ni ficheros de datos. En el directorio se encuentran los ficheros de entrada, los script y la lista de clases.
2. Proceso de ejecución de la red neuronal: Se han creado los cortes y los ficheros de datos y se puede abrir un proceso para crear los modelos.
3. Fin de la ejecución: Se han generado los ficheros con los resultados.
4. Descarga de ficheros de resultados y limpieza del directorio de trabajo: Una vez acabados todos los procesos de ejecución es necesario descargar los ficheros con los resultados y limpiar el directorio de trabajo.

A continuación, se muestra un esquema con el proceso de ejecución y los comandos ejecutados:

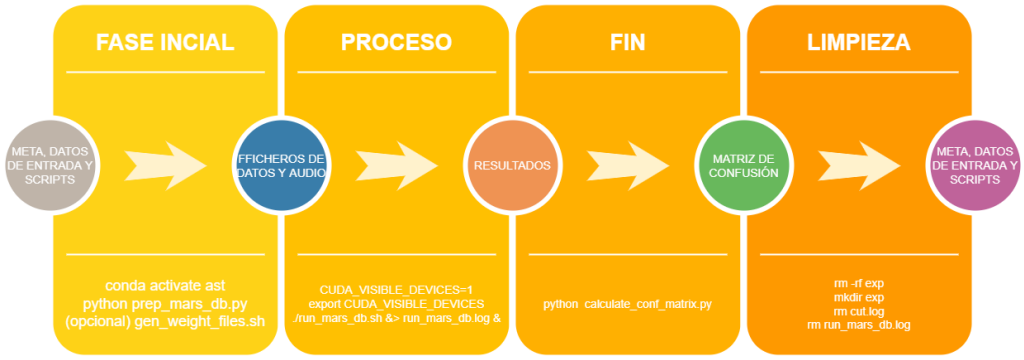

Ilustración 22 – Proceso de ejecución de la red neuronal

La carpeta *input_files* ha sido entregada con el trabajo. Aun así, se ha desarrollado un script que permite crearla descargando los ficheros del sitio de la NASA (10) y por esta razón se ha marcado como opcional en la tabla. Se puede ver una descripción de este script en el capítulo 'Script get_input_files.py'.

En la fase inicial la estructura de directorios es la siguiente:

Fichero	Descripción
🗁data	Datos de entrada de la red neuronal.
🗎 class_labels_indices.csv	Tabla con la lista completa de clases.
🗎 class_labels_indices_binary.csv	Tabla con la lista de clases para la ejecución binaria.
🗁input_data	(opcional) Ficheros originales con los eventos de sonido.
🗁meta	Carpeta con la base de datos con los eventos de sonido.
🗎 Events.csv	Hoja *events* exportada a formato *csv*.

Fichero	Descripción
Events.xlsx	Base de datos en formato Excel.
utils	Carpeta con la base de datos con los eventos de sonido.
utils.py	Funciones utilizadas por el script *prep_mars_db.py*
calculate_conf_matrix.py	Calcula las matrices de confusión después de la ejecución.
gen_weight_file.py	Script que genera un fichero de peso para la ejecución balanceada.
gen_weight_files.sh	Genera todos los ficheros de peso para la ejecución balanceada.
get_mars_db.py	Script que genera los resultados después de la ejecución.
get_input_files.py	Descarga y genera los ficheros *wav* con las etiquetas.
prep_mars_db.py	Script que genera los datos de entrada de la red neuronal.
run_mars_db.sh	Script que ejecuta la red neuronal mediante el comanda run.py.

Tabla 16 – Estructura de directorios en la fase inicial

Es necesario indicar que para la ejecución de la red neuronal el script *run.py* del proyecto *ast* debe estar colocado en el path *./../src/run.py*.

Para crear los ficheros con la información de entrada para la red neuronal es necesario seleccionar el entorno *Conda* (44) y ejecutar los siguientes comandos:

```
conda activate ast
python prep_mars_db.py
(opcional) gen_weight_files.sh
```

Ilustración 23 – Comandos de la fase inicial

Después de la ejecución de estos comandos se crean los siguientes ficheros:

Fichero	Descripción
data	Datos de entrada de la red neuronal.
audio	Ficheros con los cortes de audio de 1 / 2 segundos.
ascam_sol0037...01p01_L1_1.wav	Primer fichero con corte de 1 / 2 segundos.
ascam_sol0037...01p01_L1_2.wav	Primer fichero con corte de 1 / 2 segundos.
...	Resto de ficheros.
ascam_sol0297_01p01_SC3_12.wav	Último fichero con corte de 1 / 2 segundos.
datafiles-gr	
eval_data_1.json	Fichero de evaluación del folder 1.
eval_data_1_weight.csv	(opcional) Fichero de pesos eval_data_1.json.
eval_data_2.json	Fichero de evaluación del folder 2.
eval_data_2_weight.csv	(opcional) Fichero de pesos eval_data_2.json.
eval_data_3.json	Fichero de evaluación del folder 3.
eval_data_3_weight.csv	(opcional) Fichero de pesos eval_data_3.json.
eval_data_4.json	Fichero de evaluación del folder 4.
eval_data_4_weight.csv	(opcional) Fichero de pesos eval_data_4.json.
eval_data_5.json	Fichero de evaluación del folder 5.
eval_data_5_weight.csv	(opcional) Fichero de pesos eval_data_5.json.
train_data_1.json	Fichero de entrenamiento del folder 1.
train_data_1_weight.csv	(opcional) Fichero de pesos train_data_1.json.
train_data_2.json	Fichero de entrenamiento del folder 2.
train_data_2_weight.csv	(opcional) Fichero de pesos train_data_2.json.
train_data_3.json	Fichero de entrenamiento del folder 3.
train_data_3_weight.csv	(opcional) Fichero de pesos train_data_3.json.
train_data_4.json	Fichero de entrenamiento del folder 4.

Fichero	Descripción
train_data_4_weight.csv	(opcional) Fichero de pesos train_data_4.json.
train_data_5.json	Fichero de entrenamiento del folder 5.
train_data_5_weight.csv	(opcional) Fichero de pesos train_data_5.json.
datafiles-binary	
eval_data_1.json	Fichero de evaluación del folder 1.
eval_data_2.json	Fichero de evaluación del folder 2.
eval_data_3.json	Fichero de evaluación del folder 3.
eval_data_4.json	Fichero de evaluación del folder 4.
eval_data_5.json	Fichero de evaluación del folder 5.
train_data_1.json	Fichero de entrenamiento del folder 1.
train_data_2.json	Fichero de entrenamiento del folder 2.
train_data_3.json	Fichero de entrenamiento del folder 3.
train_data_4.json	Fichero de entrenamiento del folder 4.
train_data_5.json	Fichero de entrenamiento del folder 5.
rplc_class_labels_indices.csv	Tabla con la lista de clases para la ejecución binaria.
meta	Carpeta con la base de datos con los eventos de sonido.
asm.csv	Hoja *events* exportada a formato *csv*.
cut.log	Log de la ejecución de prep_mars_db.py.

Tabla 17 – Estructura de directorios después de la finalización de la fase inicial

Los ficheros marcados como (opcional) solamente son necesarios cuando la ejecución se realiza con el parámetro *bal=bal* (Ejecución equilibrada). Este parámetro se establece en el fichero *run_mars_db.sh*.

Una vez generada esta estructura de directorios se puede ejecutar la red neuronal mediante los siguientes comandos. Cabe destacar que es necesario seleccionar la tarjeta de video donde se va a ejecutar el proceso mediante la asignación de la variable de entorno *CUDA_VISIBLE_DEVICES*. En este caso, se ha seleccionado el número 1.

```
conda activate ast
CUDA_VISIBLE_DEVICES=1
export CUDA_VISIBLE_DEVICES
./run_mars_db.sh &› run_mars_db.log &
```

Ilustración 24 – Comandos para el entrenamiento de la red neuronal.

Tras la ejecución se genera el fichero *run_mars_db.log* y los siguientes ficheros:

Fichero	Descripción
exp	Directorio con los resultados de la ejecución.
test-esc50-f10-t10-impTrue-aspTrue-b12-lr1e-5	Directorio con los resultados de la ejecución
fold1	Resultados en el primer folder.
models	Modelos generados para el folder 1.
predictions	Resultados para el folder 1.
cum_predictions.csv	Predicción acumulada para los 25 *epoch*.
predictions_1.csv	Predicción para el *epoch* número 1.
predictions_2.csv	Predicción para el *epoch* número 2.
...	Resto de predicciones.
predictions_25.csv	Predicción para el *epoch* número 25.
target.csv	Resultados indicados durante la clasificación.
result.csv	Métricas obtenidas para el folder 1.
stats_1.pickle	Métricas obtenidas para el *epoch* 1 del folder 1.

Fichero	Descripción
...	Resto de métricas.
fold2	Resultados en el segundo folder.
models	Resultados en el primer folder.
predictions	Resultados para el folder 2.
cum_predictions.csv	Predicción acumulada para los 25 *epoch*.
predictions_1.csv	Predicción para el *epoch* número 1.
predictions_2.csv	Predicción para el *epoch* número 2.
...	Resto de predicciones.
predictions_25.csv	Predicción para el *epoch* número 25.
target.csv	Resultados indicados durante la clasificación.
result.csv	Métricas obtenidas para el folder 2.
stats_1.pickle	Métricas obtenidas para el *epoch* 1 del folder 2.
...	Resto de métricas.
...	Resto de folders.
fold5	Resultados para el folder 5.
models	Modelos generados para el folder 5.
predictions	Resultados para el folder 5.
cum_predictions.csv	Predicción acumulada para los 25 *epoch*.
predictions_1.csv	Predicción para el *epoch* número 1.
predictions_2.csv	Predicción para el *epoch* número 2.
...	Resto de predicciones.
predictions_25.csv	Predicción para el *epoch* número 25.
target.csv	Resultados indicados durante la clasificación.
result.csv	Métricas obtenidas para el folder 5.
stats_1.pickle	Métricas obtenidas para el *epoch* 1 del folder 5.
...	Resto de métricas.
acc_fold.csv	*Accuracy* global y para cada folder.
...	Resto de métricas.
best_result.csv	Mejor resultado.
result.csv	Resultado global para cada epoch.
calculate_conf_matrix.log	Log del el cálculo de métricas.

Tabla 18 – Estructura de directorios después de la fase de final

El siguiente paso es la ejecución del script *calculate_conf_matrix.py*, que genera los siguientes ficheros:

Fichero	Descripción
exp	Directorio con los resultados de la ejecución.
test-esc50-f10-t10-impTrue-aspTrue-b12-lr1e-5	Directorio con los resultados de la ejecución
fold1	Resultados en el primer folder.
predictions	Resultados para el folder 1.
confusion_matrix.csv	Matriz de confusión para el folder 1.
errors.csv	Muestras con error en el folder 1.
fold2	Resultados en el segundo folder.
predictions	Resultados para el folder 2.
confusion_matrix.csv	Matriz de confusión para el folder 2.
errors.csv	Muestras con error en el folder 2.
...	Resto de folders.
fold5	Resultados para el folder 5.
predictions	Resultados para el folder 5.
confusion_matrix.csv	Matriz de confusión para el folder 5.

Fichero	Descripción
errors.csv	Muestras con error en el folder 5.
auc_fold.csv	*AUC* global y para cada folder.
f1_fold.csv	Métrica F1 global y para cada folder.
precisión_fold.csv	Precisión global y para cada folder.
recall_fold.csv	*Recall* media global y para cada folder.
...	Ficheros con resultados del resto de métricas.
cm_plot_X.png	Matriz de confusión del *epoch* X.
cm_plot_normalized_X.png	Matriz de confusión del *epoch* X normalizada.
total_cm_X.csv	Matriz de confusión del *epoch* X en *csv*.
run_mars_db.log	Log de la ejecución de la red neuronal.

Tabla 19 – Estructura de directorios después de la fase de proceso

El último paso es descargar los resultados para su posterior análisis y limpiar las carpetas del directorio de trabajo mediante los siguientes comandos:

```
conda activate ast
rm -rf exp
mkdir exp
rm cut.log
rm run_mars_db.log
rm calculate_conf_matrix.log
rm -rf data/audio
rm -rf data/datafiles-gr
rm -rf data/datafiles-binary
rm -rf data/rplc_class_labels_indices.csv
```

Ilustración 25 – Comandos para la limpieza del directorio de trabajo

8.2. SCRIPT PREP_MARS_DB.PY

Como se ha indicado en el capítulo 'Preparación de Datos para la Red Neuronal', se ha desarrollado un script que genera los cortes de los ficheros de audio y los ficheros de datos necesarios para la ejecución de la red neuronal.

El script toma como argumentos de entrada:

Variable	Valor	Descripción
nb_msec	1000	Valor en milisegundos de cada corte.
addRest	True/False	Incluye las muestras no clasificadas en la clase *'rest'*.
limitRest	True/False	Si es *True*, utiliza limita el número de muestras *'rest'* a *total_nb_rest_events*.
total_nb_rest_events	500	Número máximo de muestras de tipo *'rest'* cuando la variable *limitRest* es igual a *true*.
order	True/False	Si es True, ordena las muestras juntas en cada folder.
orderFiles	True/false	Ordena los eventos antes de separarlos en carpetas.
overlap	0.1-1	Valor de solapamiento de las muestras. Normalmente, 0.5 o 1.
overlapAddRest	True/False	Solapa también los eventos de la clase resto cuando está a *True*.
balance	True/False	Si es *True*, iguala el número de muestras de cada clase duplicando registros.
balanceToSize	200-500	Tamaño en muestras de cada clase cuando *balance* es igual a *True*.
balanceToSizeRest	200-500	Tamaño en muestras de la clase *rest* cuando *balance* es igual a *True*.
clustering	True/False	Agrupa clases según lo indicado en la lista *group*.

Variable	Valor	Descripción
group	{ 9 : 2 }	Lista con la clase origen y la clase destino.

Tabla 20 – Variables configurables en el script *prep_mars_db.py*

Para realizar el agrupamiento, utilizado en los experimentos ha sido necesario establecer las siguientes variables:

Variable	Valor	Descripción
addClustering	True	Agrupa según lo indicando en la lista *group*.

group	group = {
	2 : 1, # Gust of wind (1) to Breeze (0)
	3 : 1, # Atmospheric Noise (2) to Breeze (0)
	6 : 5, # Wind + Helix (5) to Helix (4)
	8 : 7 # Wind + Moxie (7) to Moxie (6)
	}
		Lista con la clase origen y la clase destino.

Tabla 21 – Establecimiento de la lista *group* para el agrupamiento de clases

Para ejecutar la red neuronal es necesario definir los ficheros con las clases. Estos ficheros se describen a continuación:

Fichero	Entrada / Salida	Descripción
./data/class_labels_indices.csv	Entrada	Etiquetas de todas las clases encontradas.
./data/class_labels_indices_binary.csv	Entrada	Etiquetas de las clases evento / no evento.
./data/rplc_class_labels_indices.csv	Salida	Etiquetas de las clases después del *clustering*.

Tabla 22 – Ficheros de datos con las clases necesarios para la ejecución de la red

El script genera los ficheros de datos que utilizará la red para realizar la clasificación. A continuación, se muestra una tabla con el detalle de los ficheros de datos generados.

Clustering	Descripción	Descripción
Yes	./data-gr/train_data_1.json	Fichero de entrenamiento del folder 1 con *clustering*.
Yes	./data-gr/train_data_2.json	Fichero de entrenamiento del folder 2 con *clustering*.
Yes	./data-gr/train_data_3.json	Fichero de entrenamiento del folder 3 con *clustering*.
Yes	./data-gr/train_data_4.json	Fichero de entrenamiento del folder 4 con *clustering*.
Yes	./data-gr/train_data_5.json	Fichero de entrenamiento del folder 5 con *clustering*.
Yes	./data-gr/eval_data_1.json	Fichero de evaluación del folder 1 con *clustering*.
Yes	./data-gr/eval_data_2.json	Fichero de evaluación del folder 2 con *clustering*.
Yes	./data-gr/eval_data_3.json	Fichero de evaluación del folder 3 con *clustering*.
Yes	./data-gr/eval_data_4.json	Fichero de evaluación del folder 4 con *clustering*.
Yes	./data-gr/eval_data_5.json	Fichero de evaluación del folder 5 con *clustering*.
Yes / No	./data-binary/train_data_1.json	Fichero de entrenamiento del folder 1 evento / no evento.
Yes / No	./data-binary/train_data_2.json	Fichero de entrenamiento del folder 2 evento / no evento
Yes / No	./data-binary/train_data_3.json	Fichero de entrenamiento del folder 3 evento / no evento
Yes / No	./data-binary/train_data_4.json	Fichero de entrenamiento del folder 4 evento / no evento
Yes / No	./data-binary/train_data_5.json	Fichero de entrenamiento del folder 5 evento / no evento
Yes / No	./data-binary/eval_data_1.json	Fichero de evaluación del folder 1 evento / no evento
Yes / No	./data-binary/eval_data_2.json	Fichero de evaluación del folder 2 evento / no evento
Yes / No	./data-binary/eval_data_3.json	Fichero de evaluación del folder 3 evento / no evento
Yes / No	./data-binary/eval_data_4.json	Fichero de evaluación del folder 4 evento / no evento
Yes / No	./data-binary/eval_data_5.json	Fichero de evaluación del folder 5 evento / no evento
No	./data/train_data_1.json	Fichero de entrenamiento del folder 1 sin *clustering*.

Clustering	Descripción	Descripción
No	./data/train_data_2.json	Fichero de entrenamiento del folder 2 sin *clustering*.
No	./data/train_data_3.json	Fichero de entrenamiento del folder 3 sin *clustering*.
No	./data/train_data_4.json	Fichero de entrenamiento del folder 4 sin *clustering*.
No	./data/train_data_5.json	Fichero de entrenamiento del folder 5 sin *clustering*.
No	./data/eval_data_1.json	Fichero de evaluación del folder 1 sin *clustering*.
No	./data/eval_data_2.json	Fichero de evaluación del folder 2 sin *clustering*.
No	./data/eval_data_3.json	Fichero de evaluación del folder 3 sin *clustering*.
No	./data/eval_data_4.json	Fichero de evaluación del folder 4 sin *clustering*.
No	./data/eval_data_5.json	Fichero de evaluación del folder 5 sin *clustering*.

Tabla 23 – Ficheros de datos generados por el script *prep_mars_db.py*

Además, el script genera los ficheros de audio que se utilizarán posteriormente para la clasificación. A continuación, se muestra una tabla con algunos de los ficheros generados:

Fichero	Clase
ascam_sol0038_0670327831_525_ca2_scam07038_atm_____01p01_B1_1.wav	Brisa
ascam_sol0096_0675468235_497_ca2_scam09096_moxie_____01p01_G1_1.wav	Ráfaga de viento
ascam_sol0038_0670327831_525_ca2_scam07038_atm_____01p01_AN1_1.wav	Ruido atmosférico
ascam_sol0076_0673687636_279_ca2_scam01076_heli_____01p01_H1_1.wav	Helicoptero
ascam_sol0133_0678747754_326_ca2_scam08133_heli_____01p01_WH1_1.wav	Viento y Helicoptero
ascam_sol0081_0674102287_396_ca2_scam01081_moxie_____01p01_M1_1.wav	Moxie
ascam_sol0100_0675821930_467_ca2_scam02100_moxie_____01p01_WM1_1.wav	Viento y Moxie
ascam_sol0037_0670224261_699_ca0_scam01037_hedgehog_____01p01_L1_1.wav	Laser
ascam_sol0076_0673687636_279_ca2_scam01076_heli_____01p01_R1_1.wav	Resto

Tabla 24 – Ejemplo de los ficheros de audio generados por el script prep_mars_db.py

Además del nombre del fichero el fichero de datos contiene la etiqueta.

Las etiquetas utilizadas se muestran en la siguiente tabla:

Índice	Etiqueta	Descripción
0	/m/07rwj00	Breeze
1	/m/07rwj01	Gust_of_wind
2	/m/07rwj02	Atmospheric_noise
3	/m/07rwj03	Saturated
4	/m/07rwj04	Hélix
5	/m/07rwj05	Wind+Helix
6	/m/07rwj06	Moxie
7	/m/07rwj07	Wind+Moxie
8	/m/07rwj08	Laser
9	/m/07rwj09	Rest

Tabla 25 – Etiquetas utilizadas en la clasificación

A modo de ejemplo, a continuación, se muestra un fichero de datos. Como se ve cada fichero va acompañado de una etiqueta (*labels*) que indica la clase a la que pertenece.

```
{
 "data": [
  {
   "wav":
"./data/audio/ascam_sol0037_0670224261_699_ca0_scam01037_hedgehog_____
__01p01_L1_1.wav",
   "labels": "/m/07rwj08"
  },
  {
   "wav":
"./data/audio/ascam_sol0037_0670224376_725_ca0_scam01037_hedgehog_____
__02p01_L1_1.wav",
   "labels": "/m/07rwj08"
  },
```


Ilustración 26 – Ejemplo del fichero *eval_data_1.json* para la clasificación

El script genera el fichero *cut.log* con un resumen de las acciones realizadas durante su ejecución. A continuación, se muestra un ejemplo del contenido del *log*:

```
2022-09-05 10:19:15,186 - 4343 files processed!
2022-09-05 10:19:15,186 - Sorting events in order.
2022-09-05 10:19:15,218 - 4343 events, { 'laser' : 171, 'rest' : 1344, 'breeze' : 973, 'helix' :
391, 'moxie' : 1425, 'saturated' : 39 }
2022-09-05 10:19:15,219 - {'breeze': '0', 'saturated': '3', 'helix': '4', 'moxie': '6', 'laser': '8',
'rest': '9'}
2022-09-05 10:19:15,333 - fold 1: 3472 training samples, 871 test samples { 'laser' : 35,
'rest' : 269, 'breeze' : 195, 'helix' : 79, 'moxie' : 285, 'saturated' : 8 }
2022-09-05 10:19:15,381 - binary fold 1: 3472 training samples, 871 test samples {
'no_event' : 269, 'event' : 602 }
2022-09-05 10:19:15,562 - fold 2: 3474 training samples, 869 test samples { 'laser' : 34,
'rest' : 269, 'breeze' : 195, 'helix' : 78, 'moxie' : 285, 'saturated' : 8 }
2022-09-05 10:19:15,615 - binary fold 2: 3474 training samples, 869 test samples {
'no_event' : 269, 'event' : 600 }
2022-09-05 10:19:15,785 - fold 3: 3474 training samples, 869 test samples { 'laser' : 34,
'rest' : 269, 'breeze' : 195, 'helix' : 78, 'moxie' : 285, 'saturated' : 8 }
2022-09-05 10:19:15,834 - binary fold 3: 3474 training samples, 869 test samples {
'no_event' : 269, 'event' : 600 }
2022-09-05 10:19:16,023 - fold 4: 3475 training samples, 868 test samples { 'laser' : 34,
'rest' : 269, 'breeze' : 194, 'helix' : 78, 'moxie' : 285, 'saturated' : 8 }
2022-09-05 10:19:16,047 - binary fold 4: 3475 training samples, 868 test samples {
'no_event' : 269, 'event' : 599 }
2022-09-05 10:19:16,186 - fold 5: 3477 training samples, 866 test samples { 'laser' : 34,
'rest' : 268, 'breeze' : 194, 'helix' : 78, 'moxie' : 285, 'saturated' : 7 }
2022-09-05 10:19:16,234 - binary fold 5: 3477 training samples, 866 test samples {
'no_event' : 268, 'event' : 598 }
2022-09-05 10:19:16,282 - Finished ASMARS2022SB-10 Preparation
```

Ilustración 27 – Resultado de la ejecución del script *prep_mars_db.py*

Por último, se muestra un esquema con el diagrama de flujo del script.

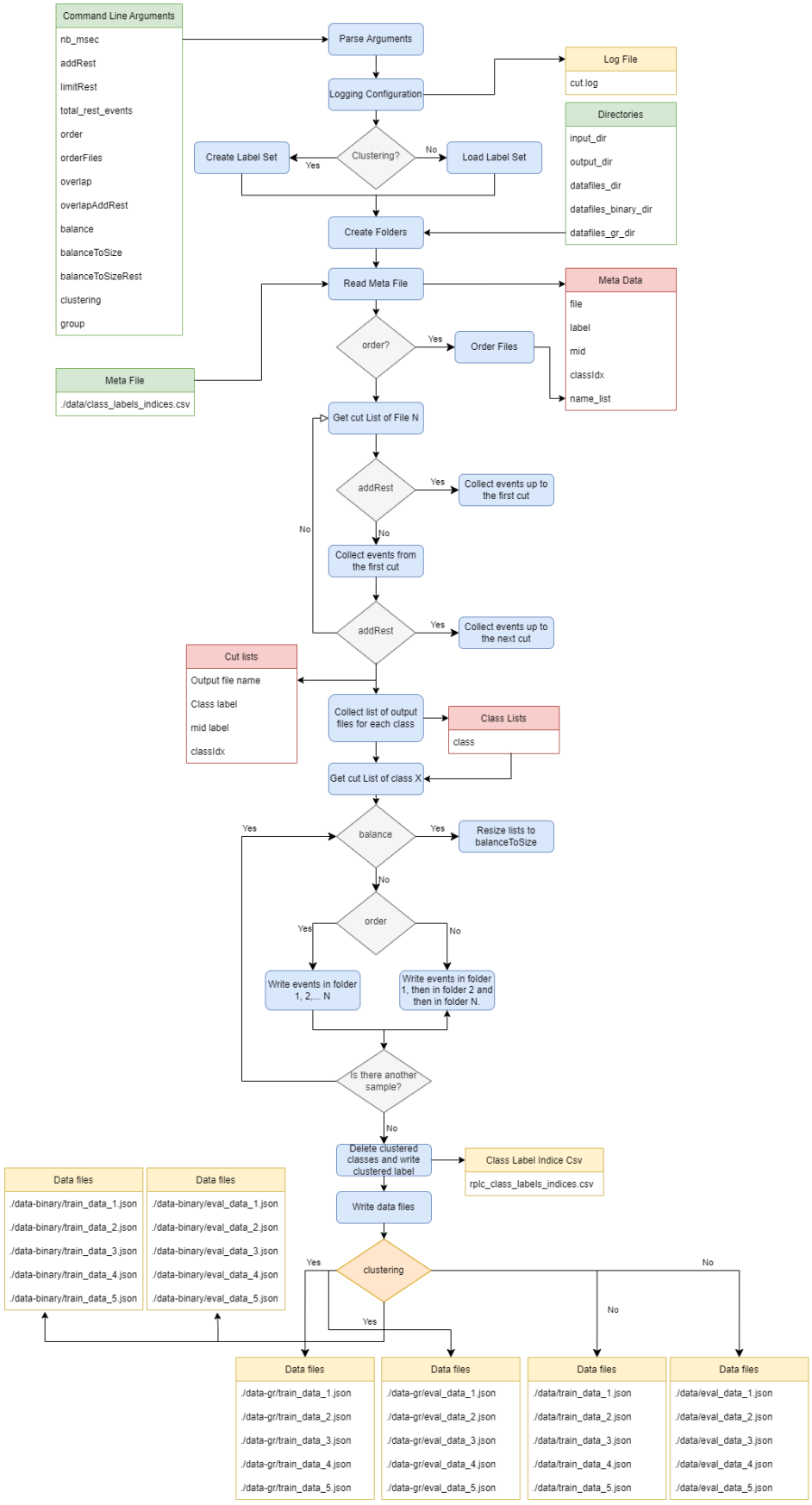

Ilustración 28 – Diagrama de flujo del script prep_mars_db.py

8.3. SCRIPT CALCULATE_CONF_MATRIX.PY

El script toma los siguientes argumentos de entrada:

Variable	Valor	Descripción
binary	True/False	Clasificación binaria.
addClustering	True/False	Indica si las clases han sido agrupadas.
averageType	'weighted'	Tipo de media utilizada en las métricas
mapType	'viridis'	Mapa de color utilizado en la matriz de confusión.
balanced	True/False	Si es *True*, calcula la métrica balanceada para la precisión.

Tabla 26 – Variables configurables en el script *calculate_conf_matrix.py*

Después de la ejecución de la red neuronal un script genera los ficheros con los resultados y las tablas de confusión. Estos ficheros se encuentran en una carpeta en la dentro del directorio *exp* cuyo nombre varía dependiendo de los parámetros de entrada de la ejecución.

Carpeta	Descripción
/exp/test-esc50-f10-t10-impTrue-aspTrue-b12-lr1e-5	Dataset=esc50
	Fstride=10
	Tstride=10
	Imagenetpretrain=True
	audiosetpretrain =True
	batch_size=12

Tabla 27 – Carpeta con los resultados de la ejecución de la red neuronal

Para la ejecución el script utiliza los siguientes ficheros de entrada:

Fichero	Descripción
/fold1/result.csv	Resultados del primer folder para cada *epoch*.
/fold2/result.csv	Resultados del segundo folder para cada *epoch*
/fold3/result.csv	Resultados del tercer folder para cada *epoch*
/fold4/result.csv	Resultados del cuarto folder para cada *epoch*
/fold5/result.csv	Resultados del quinto folder para cada *epoch*
/fold1/target.csv	Clase objetivo de cada *sample* del primer folder.
/fold2/target.csv	Clase objetivo de cada *sample* del segundo folder.
/fold3/target.csv	Clase objetivo de cada *sample* del tercer folder.
/fold4/target.csv	Clase objetivo de cada *sample* del cuarto folder.
/fold5/target.csv	Clase objetivo de cada *sample* del quinto folder.
/fold1/predictions/predictions_X.csv	Predicciones del *epoch* con el mejor resultado y el primer folder.
/fold2/predictions/predictions_X.csv	Predicciones del *epoch* con el mejor resultado y el segundo folder.
/fold3/predictions/predictions_X.csv	Predicciones para el *epoch* con el mejor resultado y el tercer folder.
/fold4/predictions/predictions_X.csv	Predicciones para el *epoch* con el mejor resultado y el cuarto folder.
/fold5/predictions/predictions_X.csv	Predicciones para el *epoch* con el mejor resultado y el quinto folder.

Tabla 28 – Ficheros con los resultados tras la ejecución de *AST*

Además, el fichero utiliza los ficheros descritos en las Tabla 23 – Ficheros de datos generados por el script prep_mars_db.py y Tabla 22 – Ficheros de datos con las clases necesarios para la ejecución de la red.

Como resultado el *script* genera los ficheros de las matrices de confusión y de los ficheros erróneos para cada folder. Además, se generan tablas de confusión con el resultado global. Estos ficheros se describen a continuación:

Fichero	Descripción
/fold1/confusion_matrix.csv	Matriz de confusión con los resultados del primer folder.
/fold2/confusion_matrix.csv	Matriz de confusión con los resultados del segundo folder.
/fold3/confusion_matrix.csv	Matriz de confusión con los resultados del tercer folder.
/fold4/confusion_matrix.csv	Matriz de confusión con los resultados del cuarto folder.
/fold5/confusion_matrix.csv	Matriz de confusión con los resultados del quinto folder.
/fold1/errors.csv	Lista de simples con resultado erróneo en el primer folder.
/fold2/errors.csv	Lista de simples con resultado erróneo en el segundo folder.
/fold3/errors.csv	Lista de simples con resultado erróneo en el tercer folder.
/fold4/errors.csv	Lista de simples con resultado erróneo en el cuarto folder.
/fold5/errors.csv	Lista de simples con resultado erróneo en el quinto folder.
total_cm_0.csv	Matriz de confusión con los resultados globales.
cm_plot_0.png	Matriz de confusión en formato png.
cm_plot_normalized_0.png	Matriz de confusión normalizada en formato png.

Tabla 29 – Ficheros con las matrices de confusión y métricas calculadas

Además, se generan ficheros con las métricas de resultados globales.

Fichero	Descripción
acc_fold.csv	Precisión (*accuracy*) media obtenida durante la ejecución.
auc_fold.csv	Área bajo la curva de características operativas del receptor (*AUC*) media obtenida durante la ejecución.
epoch_precision_fold.csv	Precisión para la mejor época.
f1_fold.csv	Valor F (*F1 score*) obtenido durante la ejecución para la mejor época,
precision_fold.csv	Precisión media obtenida durante la ejecución (utilizando todas las épocas).
recall_fold.csv	Exhaustividad (*recall*) obtenida durante la ejecución para la mejor época.

Tabla 30 – Ficheros de resultados medios para diferentes métricas

Además, el programa genera un fichero de log llamado *calculate_conf_matrix.log* con las métricas de los resultados para cada folder y los globales que se muestran a continuación a modo de ejemplo:

```
2022-08-16 12:24:31,702 - ----------Result Summary----------
2022-08-16 12:24:31,702 - Fold 1, best epoch: 4, accuracy: 0.8854
2022-08-16 12:24:31,702 - Fold 1, best epoch: 4, AUC: 0.9741
2022-08-16 12:24:31,702 - Fold 1, best epoch: 4, best epoch precision: 0.8881
2022-08-16 12:24:31,702 - Fold 1, best epoch: 4, average precision: 0.8685
2022-08-16 12:24:31,702 - Fold 1, best epoch: 4, recall score: 0.8854
2022-08-16 12:24:31,702 - Fold 1, best epoch: 4, f1 score: 0.8836
2022-08-16 12:24:31,702 - Fold 2, best epoch: 4, accuracy: 0.9120
2022-08-16 12:24:31,702 - Fold 2, best epoch: 4, AUC: 0.9776
2022-08-16 12:24:31,702 - Fold 2, best epoch: 4, best epoch precision: 0.9164
2022-08-16 12:24:31,702 - Fold 2, best epoch: 4, average precision: 0.9218
2022-08-16 12:24:31,702 - Fold 2, best epoch: 4, recall score: 0.9120
2022-08-16 12:24:31,702 - Fold 2, best epoch: 4, f1 score: 0.9124
2022-08-16 12:24:31,702 - Fold 3, best epoch: 4, accuracy: 0.8841
2022-08-16 12:24:31,702 - Fold 3, best epoch: 4, AUC: 0.9882
```

```
2022-08-16 12:24:31,702 - Fold 3, best epoch: 4, best epoch precision: 0.8999
2022-08-16 12:24:31,702 - Fold 3, best epoch: 4, average precision: 0.8646
2022-08-16 12:24:31,702 - Fold 3, best epoch: 4, recall score: 0.8841
2022-08-16 12:24:31,702 - Fold 3, best epoch: 4, f1 score: 0.8855
2022-08-16 12:24:31,702 - Fold 4, best epoch: 4, accuracy: 0.9012
2022-08-16 12:24:31,702 - Fold 4, best epoch: 4, AUC: 0.9923
2022-08-16 12:24:31,702 - Fold 4, best epoch: 4, best epoch precision: 0.9061
2022-08-16 12:24:31,702 - Fold 4, best epoch: 4, average precision: 0.8681
2022-08-16 12:24:31,702 - Fold 4, best epoch: 4, recall score: 0.9012
2022-08-16 12:24:31,714 - Fold 4, best epoch: 4, f1 score: 0.9011
2022-08-16 12:24:31,714 - Fold 5, best epoch: 4, accuracy: 0.9537
2022-08-16 12:24:31,715 - Fold 5, best epoch: 4, AUC: 0.9899
2022-08-16 12:24:31,715 - Fold 5, best epoch: 4, best epoch precision: 0.9565
2022-08-16 12:24:31,715 - Fold 5, best epoch: 4, average precision: 0.9397
2022-08-16 12:24:31,715 - Fold 5, best epoch: 4, recall score: 0.9537
2022-08-16 12:24:31,715 - Fold 5, best epoch: 4, f1 score: 0.9533
2022-08-16 12:24:31,715 - The averaged accuracy of 5 folds is 0.907
2022-08-16 12:24:31,715 - The averaged AUC of 5 folds is 0.984
2022-08-16 12:24:31,716 - The average precision of 5 folds is 0.893
2022-08-16 12:24:31,717 - The best epoch precision of 5 folds is 0.913
2022-08-16 12:24:31,717 - The recall score of 5 folds is 0.907
2022-08-16 12:24:31,719 - The f1 score of 5 folds is 0.907
```

Ilustración 29 – Ejemplo del fichero *calculate_conf_matrix.log*

Por último, se muestra un esquema con el diagrama de flujo del script.

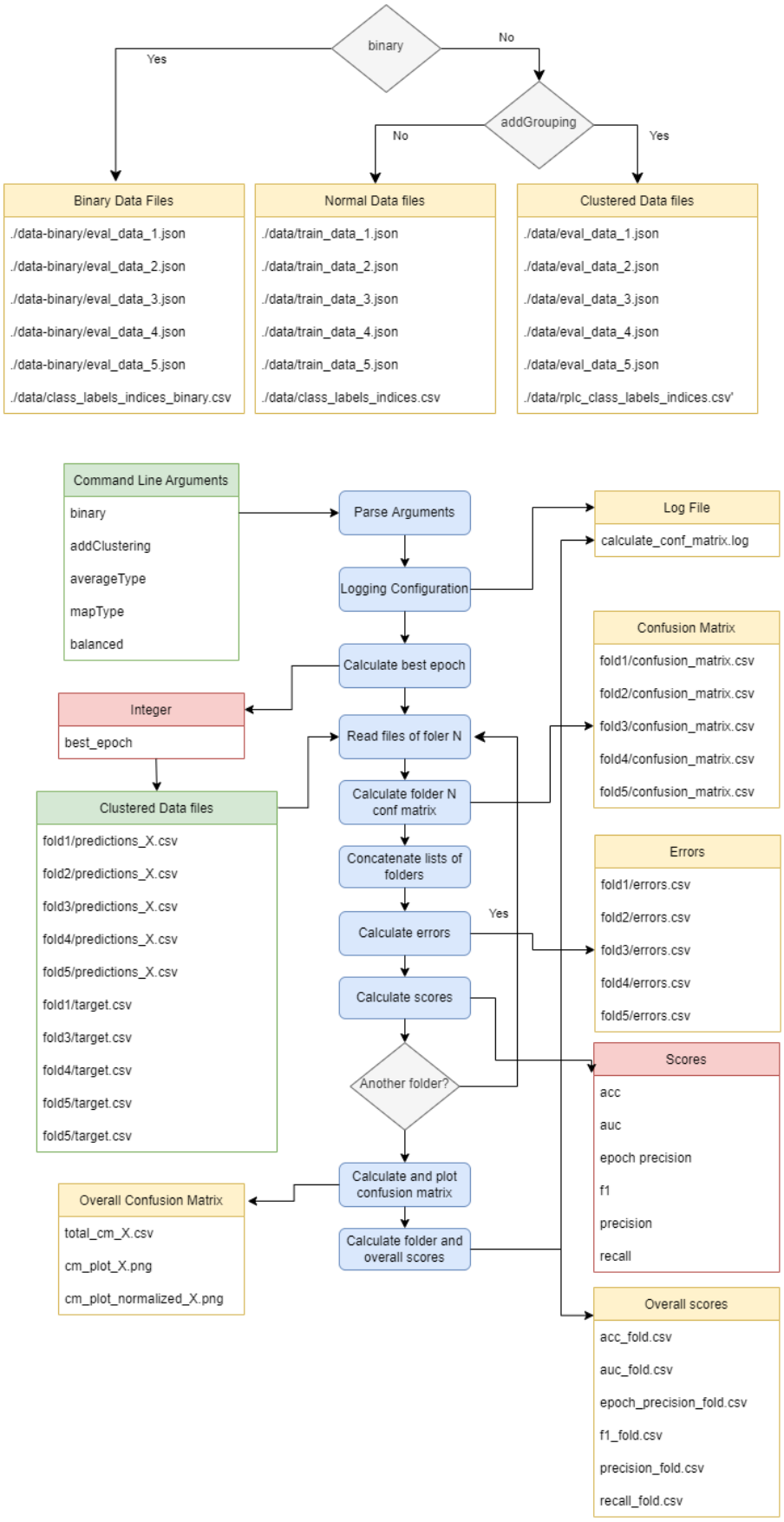

Ilustración 30 – Diagrama de flujo del script *calculate_conf_matrix.py*

8.4. SCRIPT RUN_MARS_DB.SH

Este script es el encargado de ejecutar el comando *run.py* del proyecto *ast* que se puede descargar de la siguiente url (12).

https://github.com/YuanGongND/ast

Antes de la ejecución es necesario establecer las siguientes variables:

Variable	Valores	Descripción
model	Ast	Modelo utilizado en la red.
dataset	Esc50	*Datashet* utilizado.
imagenetpretrain	True/False	Si es *True*, utiliza el modelo preentrenado de *ImageNet*.
audiosetpretrain	True/False	Si es *True*, se utiliza el modelo preformado completo de *AudioSet* e *ImageNet*.
bal	bal/False	Ejecución balanceada.
binary	False/False	Utiliza los ficheros de datos de la clasificación binaria.
addClustering	True/False	Utiliza los ficheros de datos de la clasificación agrupada.
freqm	24	Longitud máxima de la máscara de frecuencia.
timem	96	Longitud máxima de la máscara de tiempo.
mixup	0	Cuántas muestras (0-1) hay que mezclar durante el entrenamiento.
epoch	25	Número de *epoch* del entrenamiento de cada folder.
batch_size	12	Número de muestras de entrenamiento utilizados en una iteración.
fstride	10	El *stride* de la división del parche en la dimensión de la frecuencia, para 16*16 parches, *fstride*=16 significa que no hay solapamiento, *fstride*=10 significa un solapamiento de 6 (utilizado en el artículo).
Tstride	10	El *stride* de la división de parches en la dimensión temporal, para 16*16 parches, *tstride*=16 significa que no hay solapamiento, *tstride*=10 significa un solapamiento de 6 (utilizado en el documento). (por defecto:10)
Class_nb	2/7/12	Número de clases diferenciadas.

Tabla 31 – Parámetros configurable en el script *run_mars_db.sh*

Se ha ajustado el valor de procesos simultáneos *batch*_size a 12 para ocupar por lo menos 11 GB de la GPU donde se ejecutan los trabajos. Después de establecer estas variables se ejecuta el siguiente comando:

```
 CUDA_CACHE_DISABLE=1 python -W ignore ../../src/run.py --model ${model} --
dataset ${dataset} \
 --data-train ${tr_data} --data-val ${te_data} --exp-dir $exp_dir \
 --label-csv ${class_file} --n_class ${class_nb} \
 --lr $lr --n-epochs ${epoch} --batch-size $batch_size --save_model False \
 --freqm $freqm --timem $timem --mixup ${mixup} --bal ${bal} \
 --tstride $tstride --fstride $fstride --imagenet_pretrain $imagenetpretrain --
audioset_pretrain $audiosetpretrain
```

Ilustración 31 – Llamada a run.py desde el fichero *run_mars_db.sh*

Una vez ejecutado llama automáticamente a *get_mars_db.py* y *calculate_conf_matrix.py*.

8.5. SCRIPT GET_MARS_DB.PY

Este fichero se encarga de calcular las métricas medias después de la creación de los modelos. Su contenido es el mismo que el de la receta *esc50* de *ast* y se puede descargar de la siguiente url:

https://github.com/YuanGongND/ast/blob/master/egs/esc50/get_esc_result.py

8.6. SCRIPT GEN_WEIGHT_FILE.PY

Este fichero se encarga de generar los ficheros con los pesos para la ejecución equilibrada. Este fichero forma parte del proyecto *audioset* de *ast* y se puede descargar de la siguiente URL.

https://github.com/YuanGongND/ast/blob/master/egs/audioset/gen_weight_file.py

Para su uso, es necesario editar la línea 33 del fichero para incluir el fichero con la lista de clases agrupadas o binaria, cuando no se vaya a realizar la clasificación normal.

```
29 if __name__ == '__main__':
30    args = parser.parse_args()
31    data_path = args.data_path
32
33    index_dict = make_index_dict('./data/rplc_class_labels_indices.csv)
34    label_count = np.zeros(527)
35
36    with open(data_path, 'r', encoding='utf8')as fp:
```

Ilustración 32 – Modificaciones en el fichero *gen_weight_file.py*

8.7. SCRIPT GEN_WEIGHT_FILES.SH

Este fichero llama de forma iterativo al script *gen_weight_file.py* con los ficheros de datos de cada folder como argumento. Su contenido es el siguiente:

```
#!/bin/bash
for((fold=1;fold<=5;fold++));
do
 echo 'now process fold'${fold}
 python ./gen_weight_file.py --data_path ./data/datafiles-
gr/esc_eval_data_${fold}.json
 python ./gen_weight_file.py --data_path ./data/datafiles-
gr/esc_train_data_${fold}.json
done
```

Ilustración 33 – Ejemplo del fichero *gen_weight_files.sh*

8.8. SCRIPT GET_INPUT_FILES.PY

Con el objetivo de realizar la clasificación con la información recibida sin tratar, se ha programado un script que descarga los ficheros *fits* de la base de datos y extrae la información de audio para convertirla a *wav*.

Además, el programa escribe etiquetas en los ficheros *wav*, con el principio del evento (*M1s*), el final del evento (*M1e*) o el final de un evento y el principio del otro (*M1e-M2s*). Al introducir estas etiquetas se hace más sencilla la clasificación. Para escribir estas etiquetas se ha utilizado el script *wafile enhaced* que se puede descargar de la siguiente *url* (45).

https://gist.github.com/josephernest/e3903ba30b820cd199500e50f145a11f

Además, se ha sobrescrito la función *write* de manera que permita la escritura de etiquetas y se ha introducido en el fichero *utils/utils.py*.

El uso de este script no es imprescindible puesto que se cuenta con una carpeta con ficheros etiquetados. Aun así, la precisión de la red puede ser mayor utilizando la información no tratada previamente y por esta razón este script se encuentra en la carpeta de ficheros y puede ser utilizado antes de la fase inicial.

Para su ejecución es necesario establecer los siguientes parámetros.

Variable	Valores	Descripción
useWavFiles	True/False	Utiliza los ficheros *wav* como entrada en lugar de los ficheros *fits*.
overridesamplerate	25000/10000	Especifica una frecuencia de muestreo fija para las señales extraídas.
overridegain	20/3	Especifica una ganancia para la información de audio.

Tabla 32 – Variables configurables en el script *get_input_files.py*

Cabe destacar que, aunque se utilicen los ficheros *wav* el programa extraerá de los *fits* los ficheros de la clase *laser* ya que estos no se encuentran con el formato correcto en el repositorio público. Para su ejecución el script utiliza el fichero *Events.csv* descrito anteriormente y requiere los siguientes ficheros.

Fichero	Descripción
meta	Carpeta con la base de datos con los eventos de sonido.
Events.csv	Hoja *events* exportada a formato *csv*.
Events.xlsx	Base de datos en formato *Excel*.
utils	Carpeta con funciones utilizadas por varios *scripts*.
utils.py	Funciones utilizadas por diferentes *scripts*.
wavfile.py	Librería *wavfile enhaced*.
get_input_files.py	Descarga y genera los ficheros *wav* con las etiquetas.

Tabla 33 – Estructura de directorios antes de descargar los ficheros *wav*

Tras la ejecución se genera el fichero *run_mars_db.log* con los resultados y los siguientes ficheros:

Fichero	Descripción
input_data	Carpeta con la base de datos con los fragmentos de sonido.
ascam_sol0037_....wav	Primer fichero en formato *wav* convertido.
..	...
ascam_sol0297_....wav	Último fichero en formato *wav* convertido.
fits_files	Carpeta con los ficheros *fits* descargados del sito de la NASA (10).
scam_0037_...fits	Primer fichero *fits* descargado.
	..
scam_0297_....fits	Último fichero *fits* descargado.
wav_files	Carpeta con los ficheros *wav* descargados del sitio de la NASA (10).
scam_0037_...fits	Primer fichero *wav* descargado.
	..
scam_0297_....fits	Último fichero *wav* descargado.
get_input_files.log	Log con los resultados de la ejecución de get_input_files.py

Tabla 34 – Estructura de directorios después de descargar los ficheros *wav*