GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

PREDICCIÓN DE SONIDOS USANDO HARDWARE

Autor: Blanca Lluch Ponce

Director: Marc Pomar Torres

Madrid, 2020

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Predicción de Sonidos Usando Hardware

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2019/2020 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Blanca Lluch Ponce Fecha: 25/08/2020

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Marc Pomar Torres Fecha: 25/08/ 2020

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

PREDICCIÓN DE SONIDOS USANDO HARDWARE

Autor: Blanca Lluch Ponce

Director: Marc Pomar Torres

Madrid, 2020

Agradecimientos

A mi familia y amigos por su apoyo constante e incondicional.

Predicción de Sonidos Usando Hardware
Autor: Lluch Ponce, Blanca.
Director: Pomar Torres, Marc.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Es un proyecto en el que se pretende clasificar sonidos en distintos ámbitos para poder automatizar procesos. Es abierto a muchas variables ya que se podría imitar desde una casa domótica hasta un sistema de ayuda para personas con deficiencia auditiva.
Se han entrenado una red neuronal y modelo de clasificación Random Forest. Funciona a través de una API desarrollada con Keras.
Palabras clave: Python, Tensorflow, Red Neuronal Convolucional, Clasificación, Random Forest, API, Keras

1. Introducción

Una casa domótica, también conocida como ‘smart home’ (casa inteligente), es aquella en la que toda la casa puede estar controlada y automatizada por un dispositivo (ordenador o móvil) a través de internet. Esto aporta seguridad, comodidad y eficiencia energética a la casa. Se pueden controlar desde sistemas de audio, a sistemas de ventilación o iluminación, o incluso electrodomésticos de cocina. [1]

Estos sistemas pueden ser controlados gracias a ‘Internet of Things’ (IoT) es decir, “Internet de las Cosas”. Esto es la conexión entre dispositivos, a través de una red sin que requiera de la intervención de una persona. Por ejemplo, el sistema de ventilación avisando al dueño de la casa a través de su móvil, de que ha bajado la temperatura de la casa por alguna avería, o un sistema de seguridad avisando de que hay alguien en casa, o incluso un frigorífico avisando al usuario de que un alimento se está acabando o caducando. En estos casos estarían conectándose los dispositivos que detectan el problema al móvil del usuario. El objetivo principal es conseguir que los dispositivos interactúen entre ellos de manera independiente a las personas.

Internet ha avanzado mucho en los últimos años y esto ha conseguido que IoT se pueda aplicar en casi cualquier ámbito. [2] Estas aplicaciones tienen muchas ventajas pero también varios problemas que se encuentran por el camino.

Algunas de las ventajas que se presentan con IoT es que es una forma de automatizar muchos procesos, interactuar y comunicarse constantemente con la casa y conseguir así una nueva forma de control del usuario sobre sus entornos. [3] Por ello, ofrece confort y seguridad, ya que el usuario puede acceder a su casa con su móvil desde cualquier sitio y así poder saber que está ocurriendo en ella. Otra de las ventajas más interesantes es el ahorro energético y por lo tanto económico que se puede conseguir a largo plazo automatizando el control de la iluminación y la ventilación. De esta forma mediante sensores de movimiento, por ejemplo, se podría apagar la luz de una sala cuando detecte que no hay nadie. [1]

2. Definición del Proyecto

El proyecto de centra en clasificar sonidos de distintos ámbitos para poder automatizar procesos. Se entrena una red neuronal convolucional de 11 capas para que saque los parámetros de cada audio con los que posteriormente se entrena un modelo de clasificación.

Se ejecuta todo desde una API desarrollada en html y java script servida desde Flask.

3. Descripción del modelo

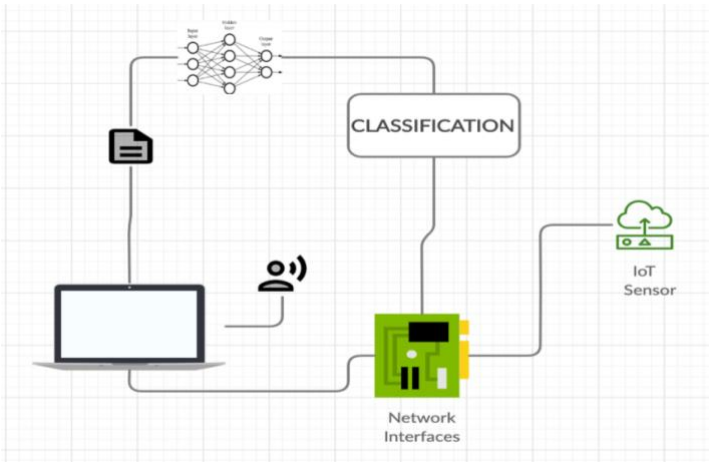


Ilustración 1 - Esquema del sistema de predicción de sonidos usando harware

4. Resultados

El entrenamiento del modelo de clasificación *Random Forest* lleva alrededor de 7 minutos con la base de datos con menor número de datos. Intentándolo entrenar con la base de datos mayor, no permite entrenar el modelo offline.

Existen dos maneras de entrenar un modelo, *offline* y *online*. La gran diferencia entre estas dos es que *offline* carga todos los datos y los entrena y la otra carga los datos poco a poco para entrenarlos. El modelo de entrenamiento (*Random Forest*) utilizado para el entrenamiento de este trabajo es de la librería *Scikit-learn,* donde solo existe el entrenamiento offline.

Para poder entrenar un tamaño tan grande de datos, es necesario utilizar un modelo en línea (*online*) con el que se consigue que se vayan cargando los datos poco a poco y que además el modelo se pueda actualizar automáticamente conforme va variando esa base de datos con la que se entrena.

La tabla 1 muestra los resultados de la predicción del modelo de clasificación, y la ilustración 2 muestra el resultado visual de la interfaz.

Modelo	mAP	AUC	mF1
Random Forest	0.453	0.735	0.631

Tabla 1 – Resultados de la predicción del modelo

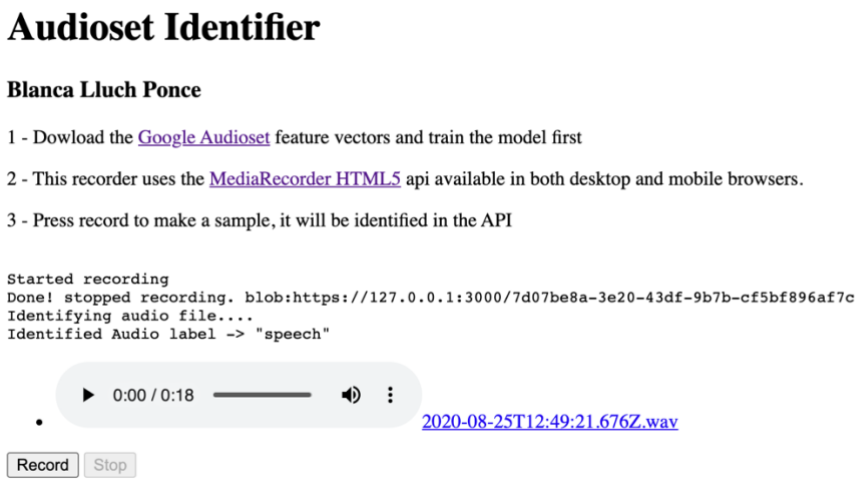


Ilustración 2 - Interfaz

5. Conclusiones

Analizando los resultados se puede comprobar que entrenando un modelo de clasificación *Random Forest*, no se aleja mucho de los resultados obtenidos por una red neuronal. La precisión media obtenida por el modelo *Random Forest* es incluso un poco mayor que la precisión media de las redes neuronales. Por otro lado, el área bajo la curva (AUC) es menor.

SI el AUC fuese 1, esto implicaría que el modelo diferencia a la perfección entre las diferentes clases. Que sea 0.735 significa que existe un 73.5% de probabilidad de que el modelo distinga entre las distintas clases. [5]

Entrenar con más datos ayuda a que el modelo aprenda más pero no por ello mejor. Puede ocurrir tanto sobreajuste (*overfitting)* como infraajuste (*underfitting*) en un modelo de aprendizaje automático. Si no hay suficientes datos el modelo no se ajusta lo suficiente a los datos, como podría ocurrir en el caso de este proyecto. Pero si ocurre un sobreajuste esto implica que el modelo es muy complejo y que ha aprendido muchos de los datos de memoria, por ello se debería probar a entrenar con un modelo más sencillo.

En el caso de este trabajo, pueden haber ocurrido dos cosas. Puede haber ocurrido un infraajuste por falta de datos porque no se ha entrenado con la base de datos

grande, pero por otro lado el modelo puede haber sido muy complejo. Esto es porque al haber tantas clases hay muchos tipos de clases parecidas, como por ejemplo palmada y aplausos o pitido de un coche y pitido de un silbato…

6. Referencias

[1] S&P, «Smart Home: qué es una casa inteligente y cuáles son sus ventajas,» SolerPalau, 15 Abril 2019. [En línea]. Available: https://www.solerpalau.com/es-es/blog/smart-home/. [Último acceso: 20 Mayo 2020].

[2] Deloitte, «IoT - Internet fo Things,» Deloitte Spain, 08 Enero 2019. [En línea]. Available: www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html.. [Último acceso: 20 Mayo 2020].

[3] N. Rivera, «Qué es el Internet of Things y cómo cambiará nuestra vida en el futuro,» Hipertextual, 21 Junio 2015. [En línea]. Available: https://hipertextual.com/2015/06/internet-of-things. [Último acceso: 20 Mayo 2020].

[4] Desconocido, «Ventajas y desventajas de tener un hogar inteligente,» Tesy, 05 Julio 2018. [En línea]. Available: https://tesy.es/blog/ventajas-y-desventajas-hogar-inteligente/. [Último acceso: 17 Junio 2020].

SOUND PREDICTION USING HARWARE

Author: Lluch Ponce, Blanca.
Supervisor: Pomar Torres, Marc.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

A project in which it's intended to classify sounds in different areas in order to automate processes. It is open to many variables since it could be from a home automation imitation to a helping system for people with hearing disability.

A neural network and Random Forest classification model have been trained for this project. It works through an API developed with Keras.

Keywords: Python, Tensorflow, Convolutional Neural Network, Classification, Random Forest, API, Keras.

1. Introduction

A home automation house, also known as a smart home, is where the entire house can be controlled and automated by a device (computer or mobile) over the internet. This brings safety, comfort and energy efficiency to the house. They can be controlled from audio systems, to ventilation or lighting systems, or even kitchen appliances.[1]

These systems can be controlled thanks to the "Internet of Things" (IoT). This is the connection between devices, through a network without requiring to the intervention of a person. For example, the ventilation system notifying the owner of the house through his mobile, that the temperature of the house has dropped due to a breakdown, or a security system warning that someone is at home, or even a refrigerator warning to the user that a food is running out or expiring. In these cases, the devices that detect the problem would be connecting to the user's mobile. The main objective is to get devices to interact with each other independently of people.

The Internet has come a long way in recent years and this has made IoT applicable in almost any field. [2] These applications have many advantages but also several problems that lie along the way.

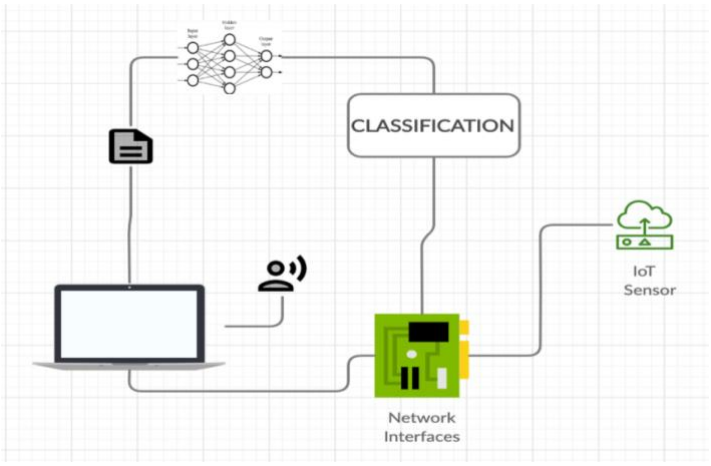
Some of the advantages that are presented with IoT is that it's a way to automate many processes, interact and communicate constantly with the house and thus achieve a new form of user control over their environments. [3] Therefore, it offers comfort and security, since the user can access his home with his mobile from anywhere and thus be able to know what is happening in it. Another of the most interesting advantages is the energy saving and therefore economic that can be achieved in the long term by automating the control of lighting and ventilation. In this way, by means of motion sensors, for example, the light in a room could be turned off when it detects that there is no one. [one]

2. Project definition

The project focuses on classifying sounds from different areas in order to automate processes. An 11-layer convolutional neural network is trained to extract the parameters of each audio with which a classification model is subsequently trained.

It runs everything from an API developed in html and java script served from Flask.

3. Model description



Ilustration 1 – Schema of the sound prediction using hardware project

4. Results

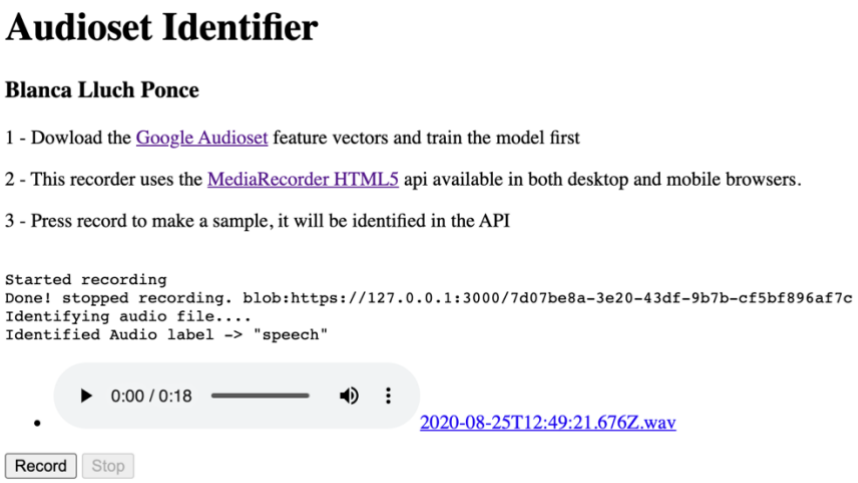
The Random Forest classification model training takes about 7 minutes with the database with the least amount of data. Trying to train with the larger database, it does not allow training the model offline. There are two ways to train a model, offline and online. The big difference between these two is that offline loads all the data and trains them. The other one loads the data little by little to train them. The training model (Random Forest) used for the training of this work is from the Scikit-learn library, where only offline training exists.

In order to train such a large size of data, it is necessary to use an online model (online) with which it is possible to load the data little by little and also that the model can be automatically updated as that database changes with which he trains.

Table 1 shows the results of the classification model prediction, and Figure 2 shows the visual result of the interface.

Modelo	mAP	AUC	mF1
Random Forest	0.453	0.735	0.631

Table 1 – Model prediction results



Ilustration 2 - Interface

5. Conclusions

Analyzing the results, it can be verified that training a Random Forest classification model does not differ much from the results obtained by a neural network. The mean precision obtained by the Random Forest model is even slightly higher than the mean precision of neural networks. On the other hand, the area under the curve (AUC) is smaller.

IF the AUC were 1, this would imply that the model differentiates perfectly between the different classes. That it is 0.735 means that there is a 73.5% probability that the model will distinguish between the different classes. [5]

Training with more data helps the model learn more but not better. Both overfitting and underfitting can occur in a machine learning model. If there is not enough data, the model does not fit the data sufficiently, as could happen in the case of this project. But if an overfitting occurs, this implies that the model is very complex and that it has learned a lot of the data from memory, so you should try to train with a simpler model.

In the case of this job, two things may have happened. An underfitting may have occurred due to lack of data because the large database has not been trained, but on the other hand the model may have been very complex. This is because, as there are so many classes, there are many types of similar classes, such as clapping and clapping or whistling a car and blowing a whistle ...

.

6. References

[1] S&P, «Smart Home: qué es una casa inteligente y cuáles son sus ventajas,» SolerPalau, 15 Abril 2019. [En línea]. Available: https://www.solerpalau.com/es-es/blog/smart-home/. [Último acceso: 20 Mayo 2020].

[2] Deloitte, «IoT - Internet fo Things,» Deloitte Spain, 08 Enero 2019. [En línea]. Available: www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html.. [Último acceso: 20 Mayo 2020].

[3] N. Rivera, «Qué es el Internet of Things y cómo cambiará nuestra vida en el futuro,» Hipertextual, 21 Junio 2015. [En línea]. Available: https://hipertextual.com/2015/06/internet-of-things. [Último acceso: 20 Mayo 2020].

[4] Desconocido, «Ventajas y desventajas de tener un hogar inteligente,» Tesy, 05 Julio 2018. [En línea]. Available: https://tesy.es/blog/ventajas-y-desventajas-hogar-inteligente/. [Último acceso: 17 Junio 2020].

[5] C. Wikipedia, «Entorno Inteligente,» Wikipedia, 03 Abril 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Entorno_inteligente. [Último acceso: 25 Mayo 2020].

Índice

Índice de Figuras

Índice de Tablas

CAPITULO 1

1.1 Introducción

Una casa domótica, también conocida como 'smart home' (casa inteligente), es aquella en la que toda la casa puede estar controlada y automatizada por un dispositivo (ordenador o móvil) a través de internet. Esto aporta seguridad, comodidad y eficiencia energética a la casa. Se pueden controlar desde sistemas de audio, a sistemas de ventilación o iluminación, o incluso electrodomésticos de cocina. [1]

Estos sistemas pueden ser controlados gracias a 'Internet of Things' (IoT) es decir, "Internet de las Cosas". Esto es la conexión entre dispositivos, a través de una red sin que requiera de la intervención de una persona. Por ejemplo, el sistema de ventilación avisando al dueño de la casa a través de su móvil, de que ha bajado la temperatura de la casa por alguna avería, o un sistema de seguridad avisando de que hay alguien en casa, o incluso un frigorífico avisando al usuario de que un alimento se está acabando o caducando. En estos casos estarían conectándose los dispositivos que detectan el problema al móvil del usuario. El objetivo principal es conseguir que los dispositivos interactúen entre ellos de manera independiente a las personas. [4]

Internet ha avanzado mucho en los últimos años y esto ha conseguido que IoT se pueda aplicar en casi cualquier ámbito. [2] Estas aplicaciones tienen muchas ventajas pero también varios problemas que se encuentran por el camino.

Algunas de las ventajas que se presentan con IoT es que es una forma de automatizar muchos procesos, interactuar y comunicarse constantemente con la casa y conseguir así una nueva forma de control del usuario sobre sus entornos. [3] Por ello, ofrece confort y seguridad, ya que el usuario puede acceder a su casa con su móvil desde cualquier sitio y así poder saber que está ocurriendo en ella. Otra de las ventajas más interesantes es el ahorro energético y por lo tanto económico que se puede conseguir a largo plazo automatizando el control de la iluminación y la ventilación. De esta forma mediante sensores de movimiento, por ejemplo, se podría apagar la luz de una sala cuando detecte que no hay nadie. [1]

Por otro lado, el problema mas grande que presentan las casas domóticas, es la elevada inversión inicial que requiere. A pesar de que cada vez haya más demanda, sigue siendo un precio inicial muy alto aunque a largo plazo se amortice. [3]

1.2 Estado de la Cuestión

La idea de un entorno o casa domótica requiere un coste elevado para la instalación porque implica un sensor específico para detectar o medir cada elemento alrededor. Existen distintos sensores, actuadores, visualizadores y elementos computacionales que se integran en los elementos del entorno y se conectan a una red, esto es lo que se llama 'Internet of things' (IoT). [5]

Este trabajo trata de conseguir una idea de automatización de algunas funciones de la casa pero a pequeña escala, únicamente mediante sensores de sonido, y sin tener que pagar un gran precio por ello.

El proyecto consiste en crear una aplicación de móvil que pueda detectar los sonidos que hay alrededor del sitio en el que esté el sensor. De esta forma, también el usuario podrá tener consigo el móvil para saber que esta ocurriendo en ese entorno.

Sería además, muy útil para personas con deficiencias auditivas que no pueden apreciar los eventos sonoros que ocurren alrededor de su casa. No obstante, este mismo sistema será usado para desencadenar acciones en un dispositivo aplicando IoT actuando así, algunos dispositivos con las personas.

1.3 Motivación

Hoy en día esta muy solicitada la automatización de muchos proceso y la tecnología esta revolucionándolo todo. Pero para poder llamar una sala 'smart home/space' o espacio inteligente, hay que comprar aparatos inteligentes que son de un coste muy elevado. ¿Por qué no mediante un hardware como un móvil, y utilizando sus propios sensores no intentamos automatizar algunos de los procesos, sin ningún coste adicional?

Esta forma además, ofrece una mayor flexibilidad porque puedes añadir más sonidos producidos por dispositivos, personas o incluso animales. En el caso de querer transformar completamente toda una casa en una casa inteligente no se podría o sería más complicado porque no existen todos los aparatos con conexión a internet, de esta forma podríamos acercarnos.

Otra necesidad de análisis de audio se habla en el posible nuevo avance de Apple [6], en la función de cancelación de ruido de los AirPods. Esta función se activa y desactiva voluntariamente, pero una posible solución para que esto no sea un peligro para el peatón, es analizar la importancia del sonido y que los AirPods determinen si activar la cancelación de ruido o no.

1.4 Objetivos del proyecto

En este proyecto se persigue resolver varios problemas.

1.4.1 Ayudar

- **Dar una recurso fácil y eficiente a las personas con deficiencias auditivas**. De esta forma podrán ser capaces de saber muchos de los sonidos que ocurren a su alrededor. Un perro ladrando en el jardín, la cafetera funcionando, la lavadora pitando porque ha terminado…

1.4.2 Reducir coste

- **Automatizar partes de un entorno sin ningún coste**. Se conseguiría utilizando un hardware de consumo que ya tiene el usuario. No deberá invertir ni en el hardware ni el los dispositivos que han de conectarse a internet para automatizar el entorno. La inversión inicial que se haría para domotizar una casa sería alrededor de unos 2.000 € . [7]

1.4.3 Flexibilidad

- **Aportar flexibilidad para poderse aplicar en distintos ámbitos.** Para poder tener un entorno inteligente se ha de comprar dispositivos inteligentes que sustituyan los que ya hay en las habitaciones. De esta forma, el hardware es el mismo y lo que se ha de hacer es entrenar la red neuronal para que detecte unos

sonidos u otros en función de lo que se desea detectar. Por ejemplo, en una cocina sería la lavadora o la cafetera encendida, la puerta de la nevera abierta y la nevera por lo tanto pitando, el grifo goteando… En cambio en una oficina podría ser la impresora funcionando, el teléfono sonando, la puerta cerrándose…

1.5 Metodología de trabajo

1.5.1 Técnicas

Este trabajo se pretende resolver mediante el entrenamiento de una **red neuronal convolucional (en ingles 'Convolutional Neural Network', CNN)** y un **modelo de clasificación** con lenguaje de programación en **Python**. El modelo de clasificación es lo que predecirá los sonidos del entorno y los clasificará desde el dispositivo. Se utilizará una topología de red neuronal para sacar las características de cada audio a analizar posteriormente.

1.5.2 Procedimiento

El primer paso es conseguir una gran base de datos con la que trabajar. Esta base de datos llamada *AudioSet Dataset* [8] contiene muchas grabaciones, sacadas de YouTube, de los sonidos que queremos predecir. El siguiente paso es extraer las características de cada audio. La base de datos contiene los audios en formato *TensorFlow Record* que consiste en un diccionario con los apartados del identificador del video de YouTube, la clase a la que pertenece, los segundos del video en el que aparece ese sonido y un vector de 128 dimensiones. Este vector de 128 dimensiones esta sacado con una red neuronal ya entrenada llamada VGGish. [9]

Una vez sacadas estas características de la red VGGish, se utilizará un modelo de clasificación llamado *Random Forest* para conseguir que realice una predicción y clasifique correctamente todos los posibles sonidos. Se creará una interfaz para interaccionar con el usuario que se podrá utilizar con un hardware de consumo. Grabará el sonido en el entorno, se extraerán las características de cada audio con la red neuronal VGGish nombrada antes y con estas características se predecirá el sonido que es. A partir de esta clasificación de sonido se accionará un disparador. Este disparador

dependerá del proceso que se quiera llevar a cabo después de la clasificación. Para avisarte cuando haya pasado una hora desde que empezó la lavadora o para guardar los tiempos en los que se suele encender una cafetera por ejemplo y poder hacer un estudio de cuanto café se necesitaría al mes. Otro ejemplo sería para desactivar la cancelación de ruido de los auriculares en el caso de escuchar un sonido peligroso que deba tener en cuenta el usuario.

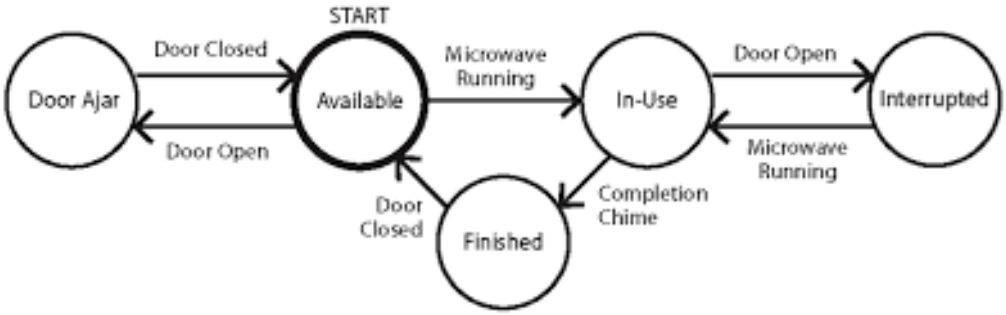


Figura 1 Ejemplo de funcionamiento del disparador [10]

1.5.3 Red neuronal Convolucional (CNN)

La red neuronal convolucional es un tipo de red neuronal. Las redes neuronales están inspiradas en el funcionamiento del cerebro humano. Aprende por si sola, comparando los resultados obtenidos con la entrada y modificándose para obtener el resultado correcto. A partir de unos valores de entrada, se genera una salida por medio de distintas capas ocultas formadas por varios nodos o neuronas conectadas entre si. Cada nodo tiene un peso y este peso es el que la red va modificando en función del resultado conseguido. [11]

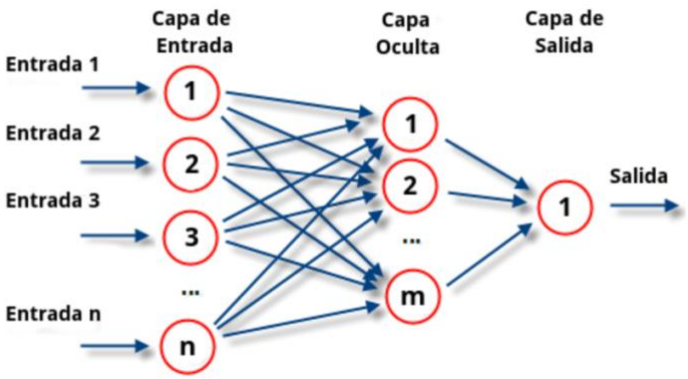


Figura 2 Funcionamiento de una Red Neuronal [11]

Cada una de estas capas esta formada por una serie de neuronas, donde cada capa se conecta a todas las neuronas de la capa anterior, como se muestra en la figura 1. En cambio, las capas de una red neuronal convolucional son de 3 dimensiones (anchura, altura y profundidad), como se muestra en la figura 2, y no está cada capa conectada a todas las neuronas de la capa anterior sino que únicamente a algunas.

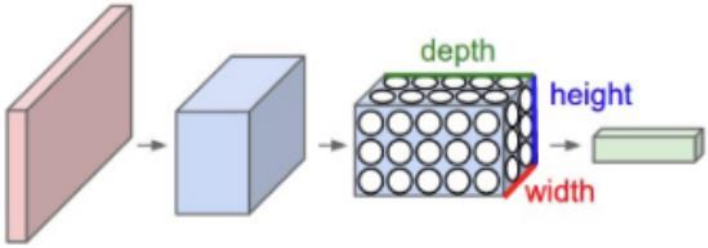


Figura 3 Estructura de una Red Neuronal Convolucional [12]

CNN

La **red neuronal convolucional** es una red muy efectiva para clasificar, que es lo necesario para este trabajo. Cada capa se entrena para que haga una tarea específica, de esta forma se reduce el tiempo de trabajo de la red porque hay menos capas ocultas. [13] Una vez analizada la entrada, se unen los datos en la capa clasificadora como se muestra en la figura 3.

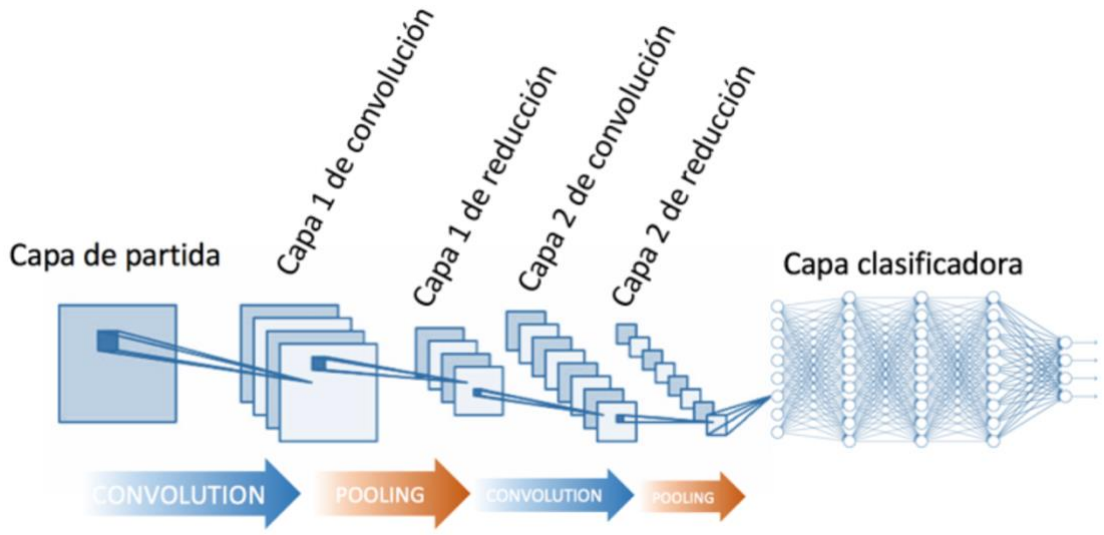


Figura 3 Arquitectura de una Red Neuronal Convolucional [13]

Capas

En la **capa de convolución** se genera un mapa de características, por medio de operaciones de productos escalares y sumas entre cada parámetro de entrada y los que tenga alrededor, y una máscara de filtrado o *kernel* como se muestra en la figura 4. [13] Mantiene la relación entre los parámetros, aplicando el mismo *kernel* a todas las matrices de la capa de partida, y sacando así la capa convolucionada en 2-dimensiones. Esta capa da la respuesta del filtro aplicado a todos los parámetros de entrada, y esto sirve para conseguir reducir el número de parámetros a entrenar. Se aplicarían tantos *kernels* como capas tenga la entrada, y por tanto la capa convolucionada tendrá este mismo número de capas.

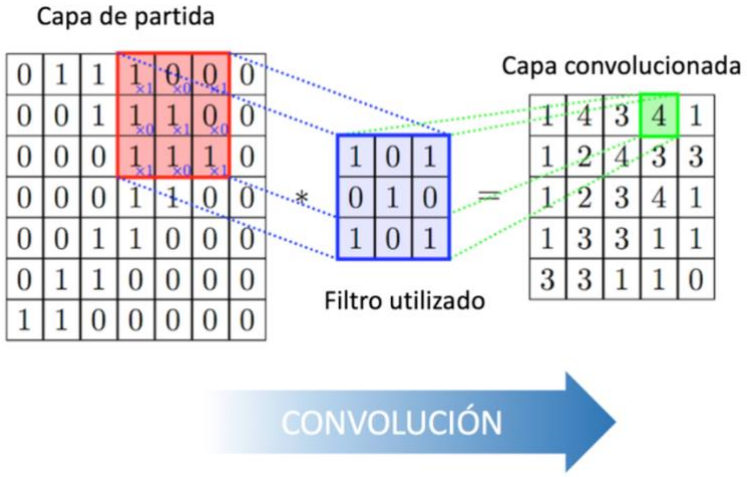


Figura 4 Capa de Convolución [13]

Una vez aplicada la convolución se aplica una capa de activación. Esta sirve para asignar los valores de un dominio a un rango de valores. Existen distintas funciones de activación pero la que se suele utilizar en CNN es ReLu, (Rectified Linear Units). [14]

La **capa de reducción** o **Pooling** consiste en extraer las características mas comunes teniendo en cuenta algunos parámetros estadísticos como son el promedio o el valor máximo. [13] Se reduce así el numero de parámetros con el que hacer la predicción pero quedándose con los valores con mayor significación. [15]

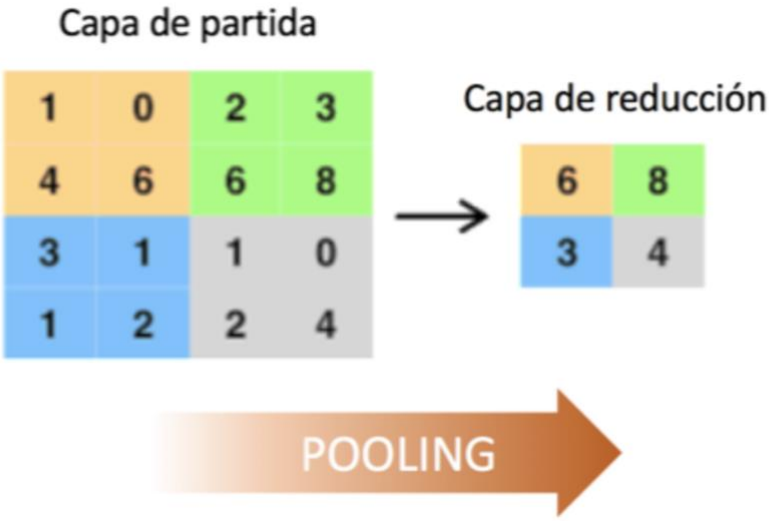


Figura 5 Capa de Reducción [13]

Por último, en la capa clasificadora la entrada que ha sido filtrada por las capas, se clasifica en varias de las posibles clases dadas para clasificar los sonidos. El vector de salida serán porcentajes de las probabilidades de que sea cada una de las clases, como se muestra en la figura 6. La capa tiene tantas neuronas como el número de clases en las que se clasifican los sonidos. Se crea un vector con los parámetros sacados a partir de las capas y este vector se clasificará utilizando una arquitectura de red. Existen varias arquitecturas importantes como son por ejemplo LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, ZFNet. [16] Estas arquitecturas de red suelen utilizarse para entrenar la clasificación de imágenes, pero muchas veces se reutiliza una misma arquitectura cambiando algunos parámetros para clasificar audios como hace referencia en este artículo académico de la Universidad de Passau, la Universidad Téctica de Múnich, la Universidad Imperial de Londres. [17]

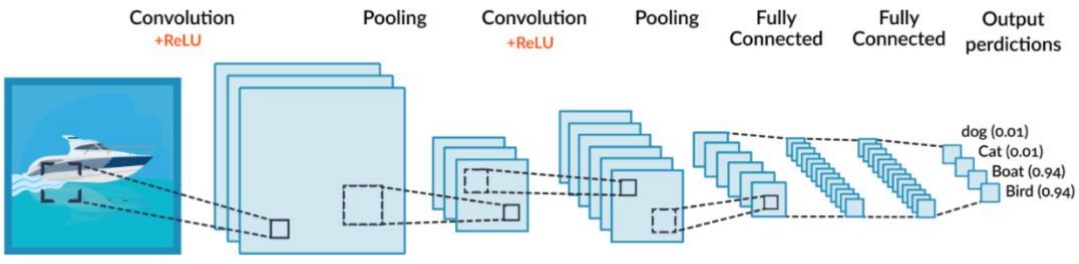


Figura 6 Red Neuronal Convolucional Completa [18]

1.6 Cronograma

	Oct	Nov	Dic	Ene	Feb	Mar	Abr	May	Jun	Jul-Ago
Lectura										
Estudio Previo										
Búsqueda de datos										
Extracción de Características de los Datos										
Tratamiento de datos										
Entrenamiento redes neuronales										
Internet of things										
Desarrollo plataforma										
Redacción										
Presentación										

Figura 7 Cronograma

1.7 Recursos a emplear

La herramienta principal a emplear es el lenguaje Python debido a la amplia oferta de librerías, y la capacidad de combinar varias de ellas para trabajar. Esto aporta al lenguaje Python una gran facilidad de uso y un rendimiento alto a la hora de trabajar con proyectos de aprendizaje automático. [19] Además, es un lenguaje muy avanzado que es aplicable a industrias diferentes como son las finanzas y la ciencia de datos.

Dentro de este lenguaje se utilizarán principalmente las librerías **TensorFlow** [20] y **Scikit-learn** [21]**. TensorFlow** es una biblioteca desarrollada por google, para trabajar con los datos utilizados en el entrenamiento y para el desarrollo y entrenamiento de la red neuronal convolucional. **Scikit-learn** es una biblioteca de aprendizaje automático para utilizar con lenguaje de programación Python que incluye algoritmos de clasificación. Los datos tratados en formato TensorFlow Record son un formato propio

de datos de TensorFlow basados en la serialización de datos de entrenamiento mediante protocolbuffers, un formato de serialización también del propio google. Para la interfaz se utilizará una web hecha en html y java script servida desde **Flask** [22] en Python. **Flask** es un entorno de trabajo desarrollado en Python que sirve para crear aplicaciones web de una manera sencilla y rápida.

1.8 Estimación Económica

Teniendo en cuenta que el lenguaje utilizado es una fuente abierta, desarrollar este trabajo no requiere de ningún coste. Todos los recursos utilizados son código abierto de gente que contribuye a la mejora de los mismos.

En el caso de querer entrenar el modelo con una base de datos mayor y necesitar entrenarlo en la nube, en la tabla 1 se muestran los posibles precios de algunas de las plataformas que ofrecen este servicio.

Plataforma	Uso	Precio
Amazon Web Service	Consumo: Para crear, entrenar e implementar modelos de aprendizaje automático.	250 horas prueba gratuita 0.84€/hora
Microsoft Azure	Windows Virtual Machines	12 meses prueba gratuita 0.3€/hora
Google Cloud	GPUs	250€ gratis 0.5€/hora

Tabla 1 Precios por uso en cada plataforma web service

CAPÍTULO 2

Procesamiento de Datos

2.1 Estructura de los Datos

Los datos con los que se trabaja son descargados de Audioset de Google [8]. Estos datos están creados por *Freesound Annotator* [23], una plataforma colaborativa abierta para crear una colección de sonidos clasificados por personas. El formato en el que están guardados estos datos es el que se muestra en la figura 8 se llama '*tensorflow.SequenceExample protocol buffers*'.

```
context: {
  feature: {
    key  : "video_id"
    value: {
      bytes_list: {
        value: [YouTube video id string]
      }
    }
  }
  feature: {
    key  : "start_time_seconds"
    value: {
      float_list: {
        value: 6.0
      }
    }
  }
  feature: {
    key  : "end_time_seconds"
    value: {
      float_list: {
        value: 16.0
      }
    }
  }
  feature: {
    key  : "labels"
      value: {
        int64_list: {
          value: [1, 522, 11, 172] # The meaning of the labels can be found here.
        }
      }
    }
}
feature_lists: {
  feature_list: {
    key  : "audio_embedding"
    value: {
      feature: {
        bytes_list: {
          value: [128 8bit quantized features]
        }
      }
      feature: {
        bytes_list: {
          value: [128 8bit quantized features]
        }
      }
    }
    ... # Repeated for every second of the segment
  }

}
```

Figura 8 Estructura de los datos de AudioSet Dataset [8]

Los audios se han sacado de Youtube, por ello entre estos datos aparece el identificador del video de Youtube, los segundos entre los que suenan los sonidos, las etiquetas de los sonidos que están sonando y por último el '*audio_embedding*' que es una lista de 128 valores por cada segundo con los que se clasifica el sonido.

2.2 Buffers de Protocolo

Tensorflow es una biblioteca de código abierto, colaborativo desarrollado por Google, para entrenar y construir redes neuronales con 'tensores' o '*arrays*' multidimensionales. [24] *Tensorflow Example* "(tf.Example o protobuf) es un tipo de mensaje flexible que representa una asignación de {'*string*':valor}" [25].

Un '*tensorflow.SequenceExample protocol buffers*' es un formato normalizado con el que se pueden guardar una secuencia de datos en formato binario para luego entrenarlos. Es una estructura flexible y compacta, ya que la puede definir el usuario y se pueden guardar de forma eficiente una gran cantidad de datos. Se requiere que los datos sean de la misma forma y por lo tanto se puedan guardar con la misma configuración. [26]

Se le llama protocol buffers o protobuf al método eficiente de serializar una estructura de datos. [27] Es muy útil para cuando se está trabajando con una gran cantidad de datos. Esto es debido a que, al ser un archivo binario, ocupa menos espacio y cargar el archivo o entrenar un modelo con él, es mucho más rápido porque al disco le cuesta menos leerlo. El inconveniente de guardar los archivos en este formato es que no existe mucha documentación para convertir la información al formato TFRecords ni para descomprimirla. [28] Un ejemplo de bufer de protocolo sería el que se muestra en la figura 9. Los datos de los protobuf pueden ser de cualquier tipo, y como se indica es con tf.train.[tipo]. El tipo de dato puede ser '*BytesList*', '*FloatList*', '*Int64List*', '*Feature*' o '*Features*', '*FeatureList*', '*FeatureLists*'… [29]

```
seq_example = tf.train.SequenceExample(
    feature_lists=tf.train.FeatureLists(
        feature_list={
            vggish_params.AUDIO_EMBEDDING_FEATURE_NAME:
                tf.train.FeatureList(
                    feature=[
                        tf.train.Feature(
                            bytes_list=tf.train.BytesList(
                                value=[embedding.tobytes()]))
                        for embedding in postprocessed_batch
                    ]
                )
        }
    )
)
```

Figura 9 Estructura de los datos del TFRecord [30]

2.3 Construcción del Dataset

Para escribir y estructurar un archivo *TFRecord* se ha de especificar como se quieren estructurar los datos. Hacen falta dos componentes: 'tf.train.Example' y 'tf.train.SequenceExample'. Se ha de guardar cada una de las muestras de los datos en una de esas dos estructuras, serializarlo y utilizar 'tf.python_io.TFRecordWriter' (escritor de *TFRecord*) para escribir y guardar los datos. [28]

Cuando se ha de leer un archivo TFRecord, han de especificarte la estructura en la cual se ha decidido guardar los datos, como lo en la figura 8. Lo primero que se ha de crear es un tf.data.TFRecordDataset(lista) de una lista de las rutas a los archivos TFRecord. Esta función devuelve un resultado que es una lista de algo parecido a la figura 10, que serían cadenas de bytes sin procesar.

```
b'\n~\n\x11\n\x08feature0\x12\x05\x1a\x03\n\x01\x00\n\x14\n\x08featu
re1\x12\x08\x12\x06\n\x04J\xa8\x1e\xbf\n\x13\n\x08feature2\x12\x07\n
\x05\n\x03dog\n>\n\x08feature3\x122\n0\n.\x08\x02\x12\x08\x12\x02\x0
8\x02\x12\x02\x08\x02" <\xec#S\x08\x9e\xd3\xbfWbu\xb16\x9e\xf3?
\xcej\xb6&\x8b$\xf1\xbf<i\x8eD\x81\t\xd2\xbf'
```

Figura 10 Resultado de un TFRecordDataset sin procesar.

Para procesar estas cadenas, se crean dos diccionarios de descripción de los datos, uno para el contexto de cada audio (*context_features)* y otro para las características o valores del audio (*features)*. Como se muestra en la figura 11, en estos dos diccionarios se indica de que tipo es cada elemento dentro del TFRecord. Estos dos diccionarios se

utilizan dentro de la función *parse_audio_embeding(example)* donde se decodifican las cadenas de bytes para conseguir un resultado más legible para poder trabajar con los datos.

```
features = {
    "audio_embedding": tf.io.FixedLenSequenceFeature([], dtype=tf.string)
}
context_features = {
    "video_id": tf.io.FixedLenFeature([], dtype=tf.string),
    "start_time_seconds":tf.io.FixedLenFeature([], dtype=tf.float32),
    "end_time_seconds":tf.io.FixedLenFeature([], dtype=tf.float32),
    "labels":tf.io.VarLenFeature(tf.int64)
}

def parse_audio_embedding(example):
    context, feats = tf.io.parse_single_sequence_example(example,sequence_features=features,
                                                         context_features=context_features)
    vals = tf.io.decode_raw(feats['audio_embedding'], tf.uint8)
    return {
        "embedding": vals,
        "context":context
    }

```

Figura 11 Función para decodificar las cadenas de bytes

CAPÍTULO 3

Entrenamiento de CNN

3.1 Entradas de CNN

La red neuronal convolucional VGGish [9] consiste en una red creada para sacar un vector de 128 dimensiones de cada segundo de un audio. Esta entrenada con los audios de AudioSet.

Estos audios están muestreados o remuestreados a 16 KHz y monoaural (mono), es decir, grabado con un único canal. De esta forma para captar la grabación únicamente hace falta un micrófono. Las características que se han utilizado para sacar las *features* de cada audio son las magnitudes de la transformada de Fourier de tiempo reducido y el espectrograma mel.

La transformada de Fourier de tiempo reducido permite determinar el contenido frecuencial en el tiempo en situaciones en las que la frecuencia de una señal, en este caso el audio, varia con el tiempo. En el caso de la transformada de Fourier estándar, se determinaría la frecuencia promedia durante todo el intervalo de tiempo. [31] Esto permite transformar una señal de audio, mostrar el contenido frecuencial en un intervalo de tiempo reducido, y poder analizarlo. Esto se consigue dividiendo la señal en ventanas y solapar una ventana con otra. Este intervalo de tiempo reducido se refleja en la figura 12, conde *Window Length* sería el tamaño de la ventana a la que aplicar la transformada de Fourier que para la red VGGish esté fijada en 25 ms, el *Hop Length* está fijado en 10ms y los 15ms restantes son el solape que aparece en la figura 12 como *Overlap Length*. [32]

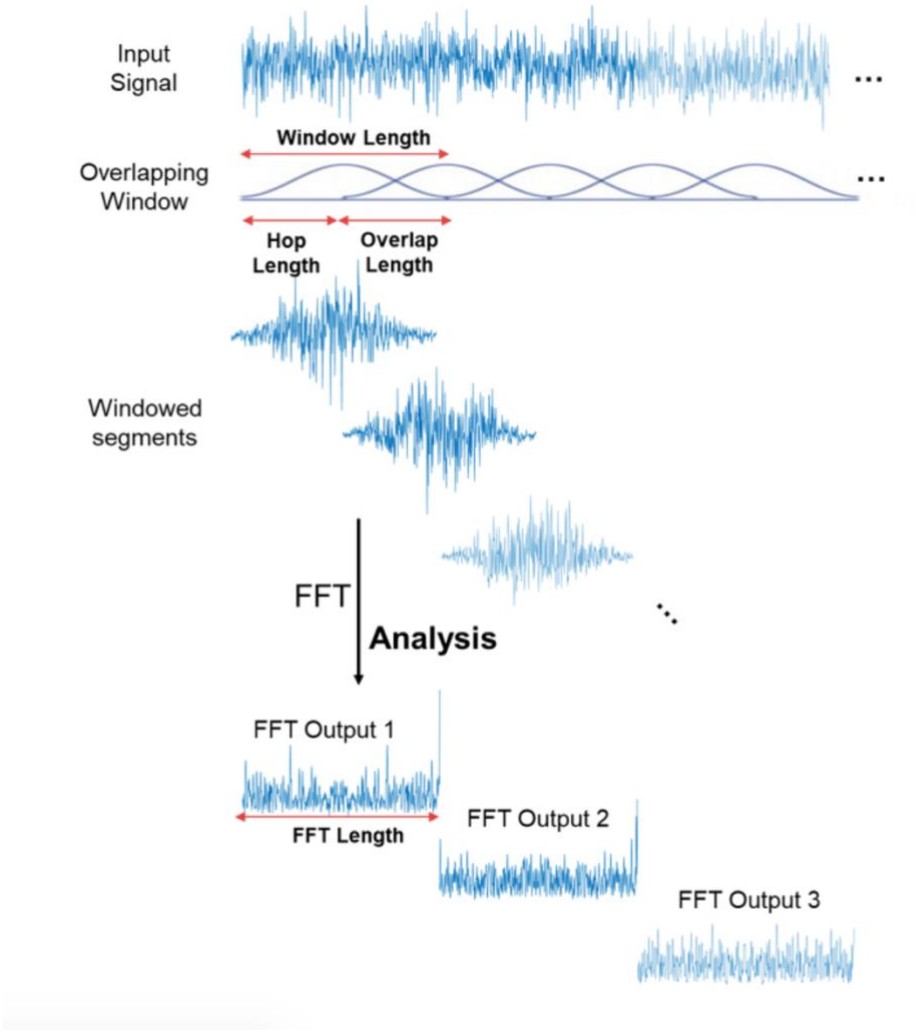


Figura 12 Transformada de Fourier de tiempo reducido [33]

El espectrograma mel se utiliza para filtrar el espectrograma de todas las magnitudes de la transformada de Fourier juntas. Esto se consigue a través de la escala Mel que aparece en la figura 13. Consiste en una unidad de todo donde las distancias iguales en el tono suenan igual de distantes para el oyente. Las personas diferencian mucho mejor los tonos en frecuencias bajas que en altas. Por ejemplo, existiendo la misma diferencia entre frecuencias, se aprecia más la diferencia entre 300Hz y 1000Hz que entre 10000Hz y 10700Hz. [32] Para las entradas de la red VGGish la escala Mel está fijada para que una escala de 125Hz a 7500Hz la divida en 64 celdas llamadas *bins*. Transformando así cada *bin* en el correspondiente *bin* de la escala Mel. [34]

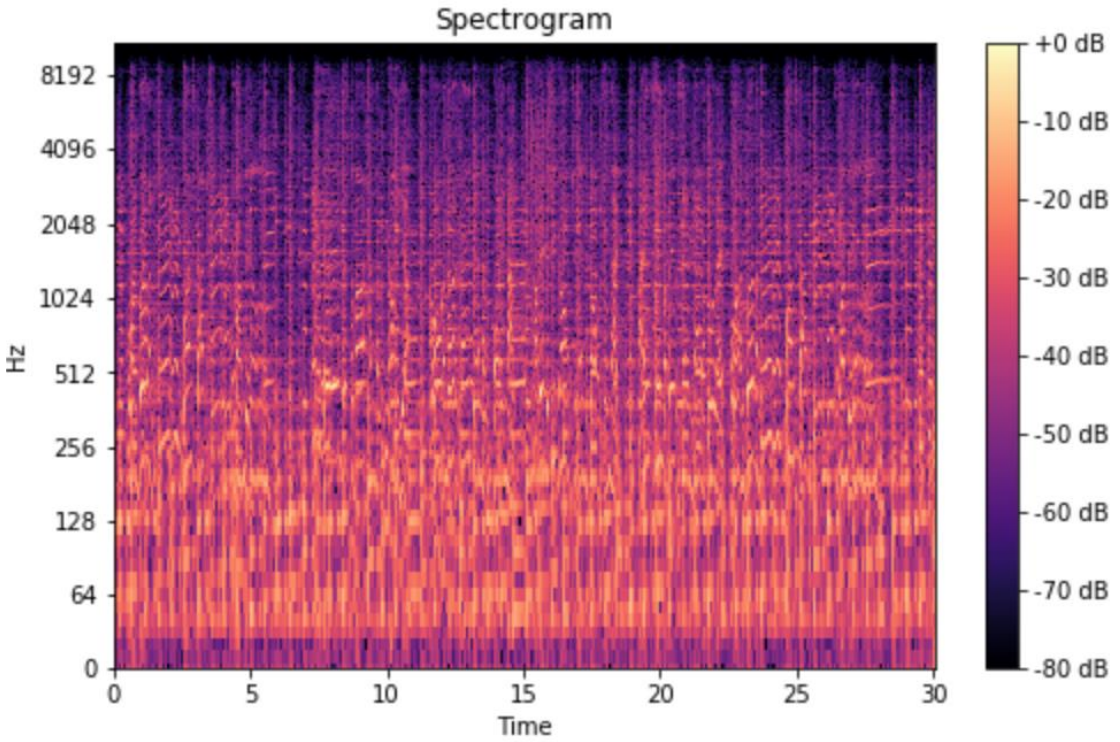


Figura 13 Espectrograma Mel [32]

Por lo tanto, la entrada de la red neuronal se trata de un audio de 1 segundo al que se le aplica una transformada de Fourier de tiempo reducido. A esta función se le aplica a continuación la función de Hann [35] o ventana Hanning. Se utiliza para el procesamiento y análisis de señales para evitar las discontinuidades al principio y al final del bloque analizado. Esto es debido a que una señal real es infinita, y al analizar únicamente una parte de esta señal, aparece mucho ruido y la ventana de Hann baja este ruido de fondo y ayuda a marcar más la salida, como se muestra en la figura 14, y que se pueda analizar mejor. [36]

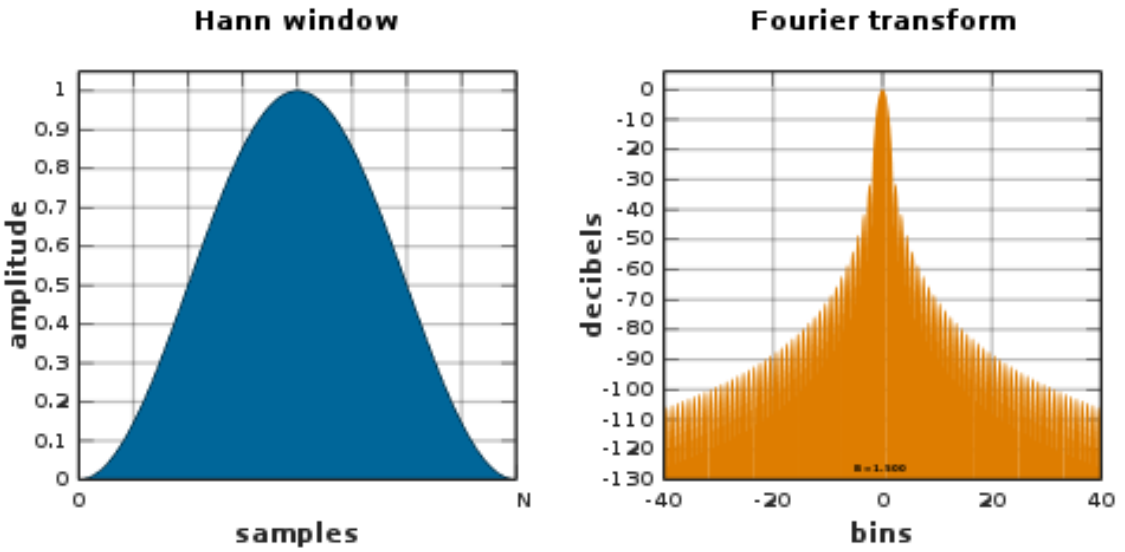


Figura 14 Ventana de Hann [35]

El siguiente filtro que se le aplica a las características antes de insertarla en la red es la magnitud del logaritmo del espectrograma mel. Se le añade un offset para evitar valores infinitos.

```
log(mel_spectrogram + offset)
```

El resultado de todas estas funciones resulta en una matriz de 96x64 (96 valores por segundo, y 64 coeficientes, uno por cada celda de el espectrograma mel) por cada segundo de audio.

3.2 Arquitectura VGGish

VGGish es una red neuronal creada con TensorFlow formada por 11 capas como se muestra en la figura 13. Estas 11 capas tienen en común varios parámetros. Entre ellos los pesos y los sesgos iniciales. Estos dos parámetros son de los mas importantes en cuanto al ritmo de aprendizaje. Al pasar un vector de salida como entrada de la siguiente capa, se le aplican a ese vector los pesos, y se pasan por una capa de activación junto con el sesgo. [37] Para los pesos se aplica un inicializador que genera valores con una distribución normal con una desviación estándar de 0.01. Para el sesgo se aplica un inicializador de ceros. Además de estos dos parámetros también tienen en común la capa de activación nombrada antes que en este caso se utiliza la capa ReLu (Rectified Linear Units). Esta consiste en aplicar los pesos y el sesgo a la salida de la capa anterior

y pasar los valores como entrada a la siguiente capa, modificándolos únicamente si son valores negativos, en este caso estos valores los pondría a cero.

```
with slim.arg_scope([slim.conv2d, slim.fully_connected],
                    weights_initializer=tf.truncated_normal_initializer(
                        stddev=params.INIT_STDDEV),
                    biases_initializer=tf.zeros_initializer(),
                    activation_fn=tf.nn.relu,
                    trainable=training), \
    slim.arg_scope([slim.conv2d],
                    kernel_size=[3, 3], stride=1, padding='SAME'), \
    slim.arg_scope([slim.max_pool2d],
                    kernel_size=[2, 2], stride=2, padding='SAME'), \
    tf.variable_scope('vggish'):
  # Input: a batch of 2-D log-mel-spectrogram patches.
  features = tf.placeholder(
      tf.float32, shape=(None, params.NUM_FRAMES, params.NUM_BANDS),
      name='input_features')
  # Reshape to 4-D so that we can convolve a batch with conv2d().
  net = tf.reshape(features, [-1, params.NUM_FRAMES, params.NUM_BANDS, 1])

  # The VGG stack of alternating convolutions and max-pools.
  net = slim.conv2d(net, 64, scope='conv1')
  net = slim.max_pool2d(net, scope='pool1')
  net = slim.conv2d(net, 128, scope='conv2')
  net = slim.max_pool2d(net, scope='pool2')
  net = slim.repeat(net, 2, slim.conv2d, 256, scope='conv3')
  net = slim.max_pool2d(net, scope='pool3')
  net = slim.repeat(net, 2, slim.conv2d, 512, scope='conv4')
  net = slim.max_pool2d(net, scope='pool4')

  # Flatten before entering fully-connected layers
  net = slim.flatten(net)
  net = slim.repeat(net, 2, slim.fully_connected, 4096, scope='fc1')
  # The embedding layer.
  net = slim.fully_connected(net, params.EMBEDDING_SIZE, scope='fc2')
```

Figura 15 CNN VGGish [30]

Entre estas 11 capas, aparecen la capa de convolución 2D, Operación de agrupación máxima 2D, repetición, aplanar y capa de conexión. Las capas de convolución 2D (conv2d) tienen un tamaño de *kernel* de 3x3, de relleno (*padding*) '*SAME'* y de zancada *(stride) 1* . El relleno '*SAME'* se utiliza para añadir ceros en los márgenes y no variar las dimensiones del vector de entrada. El efecto que tiene la zancada es sobre como aplicar el *kernel* al vector, 1 es el predeterminado. Con esta capa se genera un mapa de características, por medio de operaciones de productos escalares y sumas entre el *kernel* y los valores que este filtrando en ese momento.

Las capas de operación de agrupación máxima (max_pool2d) tienen un tamaño de *kernel* 2x2, un relleno igual que la capa de convolución y una zancada de 2. Con esta capa se elimina la influencia que pueda llegar a tener la posición de cada característica

de la entrada. Consiste en resumir los datos en el valor más activo y presente entre todos los de la entrada. [38]

Las capas de repetición (*repeat)*, aplican la misma capa con los mismos argumentos tantas veces como se le indique, en este caso 2 veces. La capa de aplanar (*flatten)* consiste en aplanar la entrada sin variar el tamaño de lote (*batch size*). El tamaño de lote consiste en el numero de elementos del *dataset* que se entrenan a la vez. El numero de iteraciones es la cantidad de lotes necesarios para completar el entrenamiento de la red con todo el *dataset*. En el caso de VGGish el *batch size* es 100. Por último, la capa de conexión (*fully_connected)* crea una matriz de pesos que multiplica por la entrada y reduce la salida al numero de características que ha de tener la salida.

En el caso de querer conectar esta red a otra para clasificar los audios sería necesario añadir una capa de activación no lineal entre las dos redes neuronales conectadas. Se utiliza una capa de activación no lineal porque produce un mayor cambio entre la salida de una red y la entrada de otra que es esencial para que la red modele y aprenda con información más compleja. [39]

3.3 Parametros VGGish

Una vez sacados los *audio embeddings* de la red neuronal se les aplica una transformación de Blanqueo con PCA (*Whitening with Principal Component Analysis*) y una cuantificación a 8 bits.

Whitening con PCA no se trata de reducir dimensiones, si no que es para transformar todos los datos en no correlacionados con misma varianza igual a 1 y media 0, normalizando así los datos. Esto es debido a que si los datos están correlacionados la clasificación sería redundante. [40]

Para aplicar la cuantificación a 8 bits en los datos, primero se eliminan los valores extremos mediante la función *clipping*. Esta función se queda con los valores entre en rango definido, en este caso entre -2 y 2. Para convertir cada *embedding* a un byte (8 bits) se reparten en un rango de 0 a 255 y de esta forma resulta cada segundo en un vector de 128 valores de 8 bits cada valor.

CAPÍTULO 4

Clasificación

4.1 Modelo de Clasificación *Random Forest*

Una vez sacadas las características de los audios a través de la red neuronal convolucional VGGish y habiendo aplicado los respectivos filtros, se han de clasificar estas características a distintos grupos. Existen varios modelos de aprendizaje automatizado (machine learning) para hacer una clasificación de multi-etiquetas, (multilabel). Entre ellos están *K Neighbors Classifier, Linear SVC (support vector classification), Decision Tree Classifier, Random Forest Classifier…*

En este estudio se va a clasificar con el clasificador *Random Forest.* Este modelo de clasificación es uno de los modelos más utilizados por su simplicidad y su buen resultado sin tener que variar mucho los parámetros. Se trata de un conjunto de árboles de decisión donde cada árbol predice una clase y la que más se repite es la clase a la que pertenecen esas características de entrada. El resultado final de una combinación de varios modelos es mejor que de un único modelo. En la figura 15 de muestra un ejemplo de lo que sería este clasificador a pequeña escala, con solo dos árboles de decisión. [41]

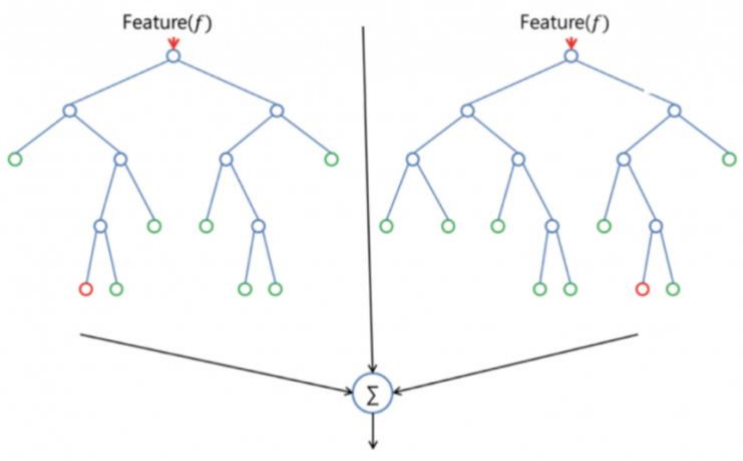


Figura 16 Ejemplo de Clasificador Random Forest [41]

El modelo Random Forest, en lugar de buscar la característica más importante entre todas ellas, busca la mejor pero solo entre un subconjunto escogido de forma aleatoria. Esto aporta más diversidad a la clasificación y da un mejor resultado.

Scikit-learn es la biblioteca que contiene el algoritmo de clasificación Random Forest. Los parámetros de este algoritmo son el numero de estimadores, es decir de arboles de decisión (fijado en 100), el número mínimo de nodos o hojas en los que se ha de separar el árbol (fijado en 2), fijando que no se utilicen todos los datos para cada árbol de decisión y que se escojan de forma aleatoria cada vez. [21]

4.2 Multi-Etiqueta

Las etiquetas de los audios a clasificar son de varías clases. En un audio aparecen varios sonidos, por ello la clasificación es multi-etiqueta (*multilabel*). Para utilizar el clasificador *random forest* se ha aplicado la función *MultiLabelBinarizer* que convierte las etiquetas a un formato legible para el modelo. Este formato es una matriz binaria donde aparece un uno en la posición igual a la etiqueta del audio, y el resto ceros.

Además, en estos audios de 10 segundos, el sonido que aparece etiquetado no suena durante los 10 segundos. No se pueden separar estos audios de un segundo en un segundo ya que en alguno de esos segundos estaría etiquetado un sonido que no aparece. Por lo tanto, se han cogido los 10 vectores de 1 segundo de audio, de 128 valores y se han juntado en un vector de 1280 valores para clasificarlo de 10 segundos en 10 segundos.

4.3 Métricas

Las métricas son necesarias para poder estudiar lo bien que predice el modelo. Todas las funciones de métricas perteneces a la librería *Scikit-learn*. Se estudiará la exactitud de la predicción del modelo con la función *accuracy_score,* siendo esto la cantidad de veces que ha acertado el modelo sobre el total de datos. La precisión con *precisión_score,* siendo la cantidad de casos verdaderos positivos entre todos los casos positivos que ha predicho el modelo. Otra métrica que se puede usar es *recall_score* que mide la cantidad de casos clasificados como verdaderos positivos sobre todo lo que

realmente (sin que el modelo lo haya predicho) era positivo. Una métrica para hacer un balance entre precisión y *recall* sería *f1-score,* el mejor valor es 1. [42]

Por otro lado, existe la curva ROC (Característica Operativa del Receptor) que se calcula con la función *roc_auc_score.* Esta curva representa el rendimiento del modelo, y la AUC es el área bajo la curva, cuanto más cercano a 1 mejor como se muestra en la figura 17, siendo TN (verdadero negativo) y TP (verdadero positivo). [43]

Otra forma de medir si un modelo de clasificación es bueno es mediante le función *average_precision_score* (precisión media). Esto se consigue calculando la precisión media por clase y luego la media de todas esas precisiones.

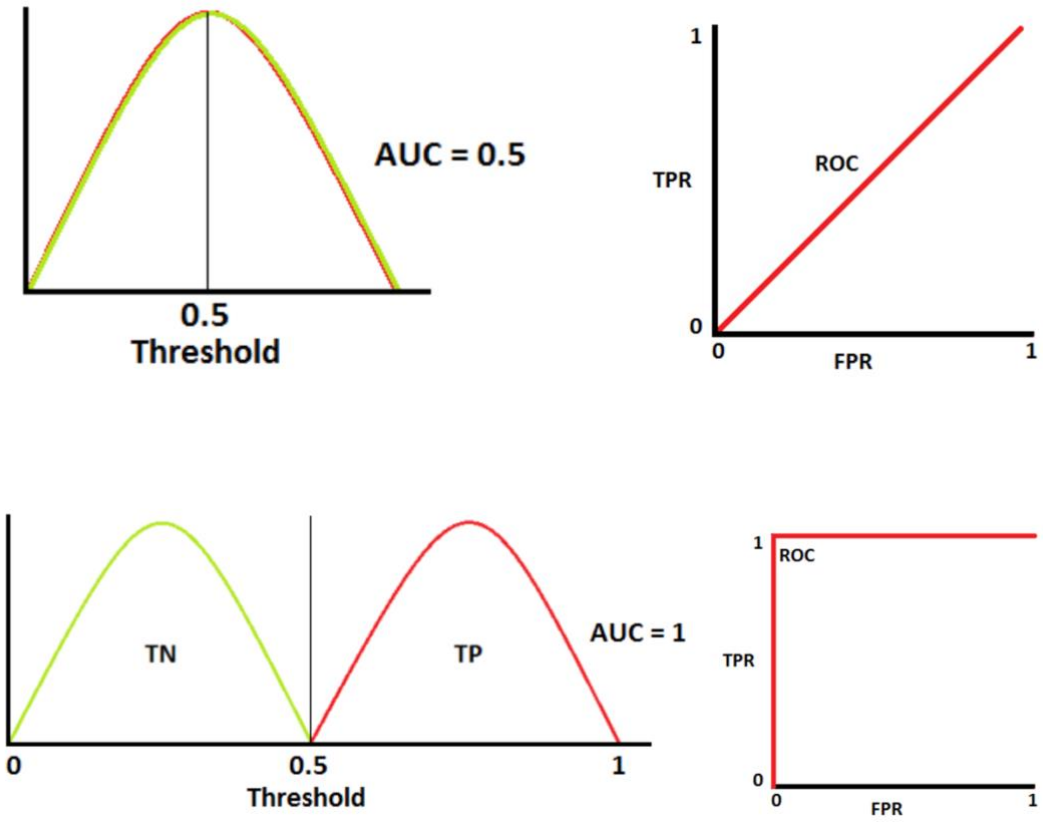


Figura 17 Curva ROC y AUC [44]

CAPÍTULO 5

Desarrollo de la API

5.1 Flask

Para el desarrollo de la interfaz de programación de la aplicación, también conocida como API se utilizará ***Flask*** [22]. Esto permite que se puedan comunicar distintos servicios y productos entre ellos. Flask esta desarrollado en Python y permite crear páginas web con un número de líneas de cogido bajo.

Al contenido o petición que un usuario manda al servidor se le llama *request.* Estas peticiones pueden ser de distintos tipos: *GET, POST, DELETE*… dependen del formato en el que envíes la petición.

5.2 Proceso

El proceso de la API consiste en grabar un audio de 10 segundos, sacar los features o características del audio, clasificarlo y devolver el resultado. Se graba el audio desde el micrófono del navegador utilizando una API de audio llamada *MediaRecorder* [45].

```
@app.route("/identify",methods=["POST"])
def identifyAudio():
    # save the file
    wav_file = request.files["blob"]
    filepath = os.path.join(app.config["upload_folder"], "test.wav")
    print(f"Storing WAV file in {filepath}")
    wav_file.save(filepath)

    print('extracting wav file')
    embedding = extractWavFile(filepath)
    print('embedding shape')
    print(embedding.shape)
    # run the inference

    print("identifying...")

    labels = identify(embedding)
    print(labels)
    if len(labels)<1:
        labels ='speech'
    data = {
        "identify":labels
    }
    return jsonify(data)
```

Figura 18 Llamando a la API MediaRecorder

Esta API es una aplicación que proporciona una sencilla forma de grabar audio desde el micrófono del navegador en lugar de grabarlo desde el ordenador. De esta forma se puede activar el micrófono desde cualquier *harware*. Se guarda la grabación en formato wav como se muestra en la figura 18 en la línea 15. Una vez guardado, se hace una petición con un *POST* comentado anteriormente. Esta petición coge el audio de 10 segundos grabado anteriormente, mete el vector del audio como entrada de la red neuronal VGGish y saca los 10 vectores de 128 dimensiones. Esta matriz será la entrada del modelo de clasificación Random Forest y la salida, que es en este caso el sonido predicho será la respuesta a la petición inicial.

La visualización de la interfaz creada con html y java script se muestra en la figura 19.

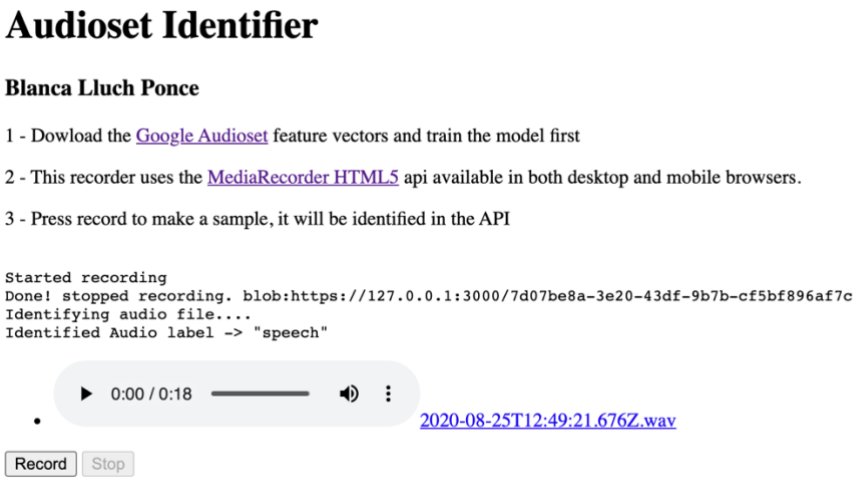


Figura 19 Interfaz

CAPÍTULO 6

Resultados

6.1 Datos

AudioSet tiene dos bases de datos [8], una llamada balanced_train_segment.csv (equilibrados) y otra llamada unbalanced_train_segment.csv (desequilibrado). La principal diferencia entre ellas es la cantidad de datos que hay, en la primera hay 12.1 MB y la segunda 1.18 GB.

El entrenamiento del modelo de clasificación *Random Forest* lleva alrededor de 7 minutos con los datos compensados (*balanced*), es decir la base de datos con menor número de datos. Intentándolo entrenar con los datos descompensados no permite entrenar el modelo offline.

Existen dos maneras de entrenar un modelo, *offline* y *online*. La gran diferencia entre estas dos es que *offline* carga todos los datos y los entrena y la otra carga los datos poco a poco para entrenarlos. El modelo de entrenamiento (*Random Forest*) utilizado para el entrenamiento de este trabajo es de la librería *Scikit-learn.* donde solo existe el entrenamiento offline.

Para poder entrenar un tamaño tan grande de datos, es necesario utilizar un modelo en línea (*online*) con el que se consigue que se vayan cargando los datos poco a poco y que además el modelo se pueda actualizar automáticamente conforme va variando esa base de datos con la que se entrena. El modelo en línea funciona debido a que aunque no se puedan comparar todos los árboles de decisión porque el modelo no tiene todos los datos de golpe, se fijan dos parámetros: la cantidad de datos que se cargan cada vez, y la importancia del árbol de decisión anterior para poderlo comparar con el nuevo. [46]

Otra forma de entrenar un modelo con una base de datos tan grande es comprar servicios informáticos en la nube como se muestra en la figura 20. Se puede trabajar

desde la nube con pases de datos o ejecutando procesos para luego interactuar con el usuario desde algún dispositivo. Esto es una forma de utilizar la GPU (unidad de procesamiento gráfico) de un ordenador más potente, desde el ordenador del usuario. Este servicio lo ofrecen muchas plataformas de distintas empresas como son Google [47], Amazon [48] y Azure [49]. Una vez entrenado el modelo desde la nube, se puede seguir entrenando con más datos (*online*) o descargar el modelo entrenado y utilizarlo desde el ordenador local (*offline*). [50]

Figura 20 Funcionamiento de servicios informáticos en la nube [51]

6.2 Métricas

Las métricas de este trabajo de clasificación se comparan con uno hecho por IBM [52]. En este caso han utilizado el conjunto de todos los datos de AudioSet [8], pero en lugar de utilizar los datos como binarios desde el archivo tensorflow, han utilizando un

ordenador con una GPU más potente, han descomprimido los datos y los han pasado a un formato hdf5 [53].

En lugar de clasificar con un modelo Random Forest, han probado a clasificar con distintas redes neuronales, variando el número de capas, el tipo de capas, tamaño del lote… Estas distintas redes neuronales se han entrenado utilizando Keras y pytorch.

Se han sacado las mismas métricas del modelo de clasificación para poder comparar los modelos. Los resultados del estudio de IBM se muestran en la figura 21 y los de la clasificación de este trabajo con el modelo Random Forest se muestra en la figura 20.

	mAP	AUC
Random guess	0.005	0.500
Google baseline	0.314	0.959
Large feature-level NN	**0.369**	**0.969**
CNN6	0.343	0.965
CNN10	0.380	0.971
CNN14	**0.431**	**0.973**
ResNet22	0.430	0.973
ResNet38	**0.434**	**0.974**
ResNet54	0.429	0.971
MobileNetV1	**0.389**	**0.970**
MobileNetV2	0.383	0.968
DaiNet	0.295	0.958
LeeNet	0.266	0.953
LeeNet18	0.336	0.963
Res1dNet30	**0.365**	**0.958**
Res1dNet44	0.355	0.948
Wavegram-CNN	0.389	0.968
Wavegram-Logmel-CNN	**0.439**	**0.973**

Figura 21 Metricas del estudio de IBM [52]

Modelo	mAP	AUC	mF1
Random Forest	0.453	0.735	0.631

Figura 22 Metricas de la clasificación Random Forest

La letra m de la métrica mAP indica el tipo de media hecha. En este caso se ha calculado la media del tipo micro que es la precisión media de todas las clases, en lugar de calcular la precisión de cada clase.

CAPÍTULO 7
Análisis de Resultados

7.1 Conclusión

Analizando los resultados se puede comprobar que entrenando un modelo de clasificación *Random Forest*, no se aleja mucho de los resultados obtenidos por una red neuronal. La precisión media obtenida por el modelo *Random Forest* es incluso un poco mayor que la precisión media de las redes neuronales. Por otro lado, el área bajo la curva (AUC) es menor.

SI el AUC fuese 1, esto implicaría que el modelo diferencia a la perfección entre las diferentes clases. Que sea 0.735 significa que existe un 73.5% de probabilidad de que el modelo distinga entre las distintas clases. [54]

Entrenar con más datos ayuda a que el modelo aprenda más pero no por ello mejor. Puede ocurrir tanto sobreajuste (*overfitting)* como infraajuste (*underfitting*) en un modelo de aprendizaje automático. Si no hay suficientes datos el modelo no se ajusta lo suficiente a los datos, como podría ocurrir en el caso de este proyecto. Pero si ocurre un sobreajuste esto implica que el modelo es muy complejo y que ha aprendido muchos de los datos de memoria, por ello se debería probar a entrenar con un modelo más sencillo.

En el caso de este trabajo, pueden haber ocurrido dos cosas. Puede haber ocurrido un infraajuste por falta de datos porque no se ha entrenado con la base de datos grande, pero por otro lado el modelo puede haber sido muy complejo. Esto es porque al haber tantas clases hay muchos tipos de clases parecidas, como por ejemplo palmada y aplausos o pitido de un coche y pitido de un silbato…

7.2 Mejoras y próximos pasos

Como posibles mejoras para continuar con el trabajo se intentarán mejorar las predicciones. Esto se hará entrenando en un ordenador con más GPU que pueda

entrenar más datos. Se probarán también más modelos de clasificación con distintos parámetros o clasificar con una red neuronal nueva, conectada con una capa de activación a la red que crea los vectores de 128 dimensiones.

Una vez mejorada la predicción, subir la interfaz creada a la nube para poder compartir el proyecto e interactuar con otros usuarios. A la predicción se le aplicara un filtro de clases con las que ha de actuar el disparador y con las que no. El disparador que se actuará dependerá del usuario.

CAPÍTULO 8
Bibliografía

[1] S&P, «Smart Home: qué es una casa inteligente y cuáles son sus ventajas,» SolerPalau, 15 Abril 2019. [En línea]. Available: https://www.solerpalau.com/es-es/blog/smart-home/. [Último acceso: 20 Mayo 2020].

[2] N. Rivera, «Qué es el Internet of Things y cómo cambiará nuestra vida en el futuro,» Hipertextual, 21 Junio 2015. [En línea]. Available: https://hipertextual.com/2015/06/internet-of-things. [Último acceso: 20 Mayo 2020].

[3] Desconocido, «Ventajas y desventajas de tener un hogar inteligente,» Tesy, 05 Julio 2018. [En línea]. Available: https://tesy.es/blog/ventajas-y-desventajas-hogar-inteligente/. [Último acceso: 17 Junio 2020].

[4] Deloitte, «IoT - Internet fo Things,» Deloitte Spain, 08 Enero 2019. [En línea]. Available: www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html.. [Último acceso: 20 Mayo 2020].

[5] C. Wikipedia, «Entorno Inteligente,» Wikipedia, 03 Abril 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Entorno_inteligente. [Último acceso: 25 Mayo 2020].

[6] M. LÓPEZ, «Apple quiere que los AirPods dejen de cancelar el sonido para alertar al usuario en caso de necesidad,» applesfera, 12 08 2020. [En línea]. Available: https://www.applesfera.com/accesorios/apple-quiere-que-airpods-dejen-cancelar-sonido-para-alertar-al-usuario-caso-necesidad. [Último acceso: 18 08 2020].

[7] C. Proyectos, «Culmen Servicios y proyectos, S.L.,» Culmen proyectos, [En línea]. Available: http://www.culmenproyectos.com/index.php?option=com_content&view=article&id=53&Itemid=57. [Último acceso: 07 07 2020].

[8] Google, «AudioSet,» Google, [En línea]. Available: https://research.google.com/audioset/download.html. [Último acceso: 23 06 2020].

[9] dpwe, «Tensorflow/Models,» GitHub, 17 Abril 2020. [En línea]. Available: https://github.com/tensorflow/models/tree/master/research/audioset. [Último acceso: 15 Junio 2020].

[10] G. Laput, Y. Zhang y C. Harrison, «Synthetic Sensors: Towards General-Purpose Sensing,» Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh.

[11] A. Innovation, «Qué son las redes neuronales y sus funciones,» ATRIA, 2019. [En línea]. Available: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/. [Último acceso: 5 Junio 2020].

[12] D. Cornelisse, «An intuitive guide to Convolutional Neural Networks,» freecodecamp, 24 Abril 2018. [En línea]. Available: https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/. [Último acceso: 20 Junio 2020].

[13] D. Calvo, «Red Neuronal Convolucional CNN,» Diego Calvo, 8 Diciembre 2018. [En línea]. Available: https://www.diegocalvo.es/red-neuronal-convolucional/. [Último acceso: 5 Junio 2020].

[14] Desc., «ReLU: Funciones de activación,» sitop big data, 22 06 2019. [En línea]. Available: https://sitiobigdata.com/2019/06/22/relu-funciones-activacion/. [Último acceso: 23 06 2020].

[15] Prabhu, «Understanding of Convolutional Neural Network (CNN) — Deep Learning,» medium, 4 03 2018. [En línea]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. [Último acceso: 29 06 2020].

[16] S. Saha, «A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,» towardsdatascience, 15 12 2018. [En línea]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Último acceso: 29 06 2020].

[17] S. S. C. U. I.-b. D. S. Features, «Google Scholar,» 24 08 2017. [En línea]. Available: https://scholar.google.es/scholar?hl=es&as_sdt=0%2C5&q=cnn+sound+classification&btnG=&oq=cnn+sound+clas. [Último acceso: 07 07 2020].

[18] Desconocido, «Convolutional Neural Network Tutorial: From Basic to Advanced,» missingink.ai, [En línea]. Available: https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/. [Último acceso: 29 06 2020].

[19] «Machine learning en python,» graph everywhere, [En línea]. Available: https://www.grapheverywhere.com/machine-learning-en-python/#:~:text=Python%20se%20caracteriza%20por%20funcionar,cient%C3%ADfico%20y%20el%20mundo%20empresarial.&text=Por%20%C3%BAltimo%2C%20es%20importante%20destacar,realizar%20proyectos%20de%20machine%20le. [Último acceso: 24 08 2020].

[20] Google, «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/. [Último acceso: 29 06 2020].

[21] «scikit-learn,» scikit-learn, [En línea]. Available: https://scikit-learn.org/stable/.

[22] «Flask web development, one drop at a time,» Flask, [En línea]. Available: https://flask.palletsprojects.com/en/1.1.x/. [Último acceso: 29 06 2020].

[23] U. P. Fabra, «Freesound Annotator, MTG,» FSC, 5 Marzo 2017. [En línea]. Available: https://annotator.freesound.org/fsd/explore/. [Último acceso: 28 Mayo 2020].

[24] «Tensorflow,» Wikipedia, 10 07 2020. [En línea]. Available: https://es.wikipedia.org/wiki/TensorFlow. [Último acceso: 13 08 2020].

[25] G. Developers, «Tensorflow,» 23 07 2020. [En línea]. Available: https://www.tensorflow.org/tutorials/load_data/tfrecord. [Último acceso: 13 08 2020].

[26] «GitHub,» GitHub, 05 02 2020. [En línea]. Available: https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/example/example.proto. [Último acceso: 13 08 2020].

[27] «Protocol Buffers,» Wikipedia, 09 08 20. [En línea]. Available: https://en.wikipedia.org/wiki/Protocol_Buffers. [Último acceso: 13 08 2020].

[28] T. Gamauf, «Tensorflow Records? What they are and how to use them,» Medium, 20 03 2018. [En línea]. Available: https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564. [Último acceso: 13 08 2020].

[29] C. Soylu, «Working with TFRecords and tf.train.Example,» towards science, 18 07 2019. [En línea]. Available: https://towardsdatascience.com/working-with-tfrecords-and-tf-train-example-36d111b3ff4d. [Último acceso: 15 08 2020].

[30] plakal, «tensorflow/models/audioset,» GitHub, 10 07 2020. [En línea]. Available: https://github.com/tensorflow/models/blob/master/research/audioset/vggish/vggish_inference_demo.py. [Último acceso: 14 07 2020].

[31] N. Kehtarnavaz, « Frequency Domain Processing,» ScienceDirect, 2008. [En línea]. Available: https://www.sciencedirect.com/topics/engineering/short-time-fourier-transform. [Último acceso: 16 08 2020].

[32] L. Roberts, « Responses (1) What are your thoughts? Cancel Respond Vimal Bhat Vimal Bhat 3 months ago Good job in explaining the concept well. Thanks for the 3Blue1Brown Fourier video. Understanding the Mel Spectrogram,» Medium, [En línea]. Available: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53. [Último acceso: 15 08 2020].

[33] MathWorks, «dsp.STFT,» MathWorks, 2020. [En línea]. Available: https://www.mathworks.com/help/dsp/ref/dsp.stft.html. [Último acceso: 16 08 2020].

[34] D. Gartzman, «Getting to Know the Mel Spectrogram,» towardsDatascience, 19 08 2019. [En línea]. Available: https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0. [Último acceso: 16 08 2020].

[35] B. K, «Hann function,» Wikipedia, 30 07 2020. [En línea]. Available: https://en.wikipedia.org/wiki/Hann_function. [Último acceso: 16 08 2020].

[36] «Ventana (función),» Wikipedia, 25 04 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Ventana_(funci%C3%B3n). [Último acceso: 16 08 2020].

[37] F. Malik, «Neural Networks Bias And Weights,» Medium, 18 05 2019. [En línea]. Available: https://medium.com/fintechexplained/neural-networks-bias-and-weights-10b53e6285da#:~:text=The%20weights%20and%20bias%20are,function%20along%20with%20the%20bias.. [Último acceso: 16 08 2020].

[38] J. Brownlee, «A Gentle Introduction to Pooling Layers for Convolutional Neural Networks,» machinelearningmastery, 22 04 2019. [En línea]. Available: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/. [Último acceso: 17 08 2020].

[39] «7 Types of Neural Network Activation Functions: How to Choose?,» Missinglink, [En línea]. Available: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/. [Último acceso: 16 08 2020].

[40] S. P. Singh, «Whitening with PCA with code demonstration,» Opengenus, [En línea]. Available: https://iq.opengenus.org/whitening-with-pca/. [Último acceso: 16 08 2020].

[41] N. Donges, «A COMPLETE GUIDE TO THE RANDOM FOREST ALGORITHM,» Builtin, 16 06 2019. [En línea]. Available: https://builtin.com/data-science/random-forest-algorithm. [Último acceso: 17 08 2020].

[42] waster, «Explicación alternativa para accuracy, precision, recall y f1-score,» steemit, 01 2020. [En línea]. Available: https://steemit.com/spanish/@waster/explicacion-alternativa-para-accuracy-precision-recall-y-f1-score. [Último acceso: 16 08 2020].

[43] «Clasificación: ROC y AUC,» developers google, 10 02 2020. [En línea]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc. [Último acceso: 17 08 2020].

[44] S. Narkhede, «Understanding AUC - ROC Curve,» towardsdatascience, 26 06 2018. [En línea]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. [Último acceso: 16 08 2020].

[45] «MediaRecorder,» developer mozilla, [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder. [Último acceso: 24 08 2020].

[46] J. K. u. S. Schmalz, «Network Anomaly Detection: Online vs. Offline Machine Learning,» inovex blog, 12 12 2017. [En línea]. Available: https://www.inovex.de/blog/online-offline-machine-learning-network-anomaly-detection/. [Último acceso: 24 08 2020].

[47] google, «Compute Engine,» google, [En línea]. Available: https://cloud.google.com/compute.

[48] «Amazon EC2,» Amazon, [En línea]. Available: https://aws.amazon.com/es/ec2/.

[49] «Azure virtual machine,» Azure Microsoft, [En línea]. Available: https://azure.microsoft.com/en-us/services/virtual-machines/.

[50] A. Amjadi, «Online vs. Offline Machine learning: Which one to use?,» good audience, 08 07 2018. [En línea]. Available: https://blog.goodaudience.com/online-vs-offline-machine-learning-which-one-to-use-934801e0a418. [Último acceso: 24 08 2020].

[51] «Posibilidades de cloud computing,» tic, [En línea]. Available: https://www.ticportal.es/temas/cloud-computing/posibilidades-cloud-computing. [Último acceso: 24 08 2020].

[52] «IBM/audioset-classification,» github, 22 04 2019. [En línea]. Available: https://github.com/IBM/audioset-classification. [Último acceso: 24 08 2020].

[53] «HDF5 Library and File Format,» HDF Group, [En línea]. Available: https://www.hdfgroup.org/solutions/hdf5/.

[54] L. Gonzalez, «Curvas ROC y Área bajo la curva (AUC),» ligdigonzalez, 31 05 2019. [En línea]. Available: https://ligdigonzalez.com/curvas-roc-y-area-bajo-la-curva-auc-machine-learning/. [Último acceso: 26 08 2020].

[55] Na8, «¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador,» aprendemachinelearning, 28 11 2018. [En línea]. Available: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/. [Último acceso: 23 06 2020].

[56] Keras, «Keras,» [En línea]. Available: https://keras.io/. [Último acceso: 29 06 2020].

[57] C. o. l. s. u. c. n. networks, «SpringerLink,» 2017. [En línea]. Available: https://link.springer.com/article/10.1186/s13640-017-0213-2. [Último acceso: 07 07 2020].

Anexo I
ODS (Objetivos de Desarrollo Sostenible)

Los objetivos de un desarrollo sostenible es una forma de obligar a todos a pensar de forma creativa para ayudar a tener un desarrollo sostenible. Las partes a desarrollar son 5: Las personas, la prosperidad, la paz, la cooperación y el planeta.

Estas partes construyen 17 objetivos sostenibles. Entre estos 17 este trabajo puede ayudar a alcanzar algunos de ellos.

Dimensión ODS	*ODS Identificado*	*Rol*	*Objetivo*
Biosfera	ODS13: Medidas contra el cambio climático	Secundario	Aportar conocimientos de la cantidad de agua que se consume siendo este un suministro básico de primera necesidad. (Grabando lo que ocurre en la casa se podría anotar la cantidad de litros que se gastan al día).
Sociedad	ODS11: Ciudades y comunidades sostenibles	Primario	La posibilidad de automatizar parte de una casa o cualquier otro entorno sin necesidad de cambiar y tirar los electrodomésticos que ya existen en ese entorno.
	ODS10: Reducción de desigualdades	Primario	Aportar a las personas con deficiencia auditiva una nueva forma de sentirse incluidos pudiendo entender y apreciar los eventos sonoros que ocurren a su alrededor.
Economía	ODS12: Producción y consumo responsable	Secundario	Aportar conocimientos de la cantidad de agua que se consume al día es un incentivo para

			conseguir una gestión sostenible y uso eficiente de los este recurso natural como es el agua.

Tabla 2 Objetivos de Desarrollo Sostenible del proyecto

Cuantificación del impacto de la contribución a los ODS

- Reducir el consumo elevado de algunos bienes en muchos entornos, ya que *'el cambio climático es un desafío para el acceso universal al agua dulce, un recurso ya de por sí sobreexplotado por el progresivo aumento de la población, la urbanización y la industrialización. '* [55]
- *Automatizar entornos sin necesidad de cambiar los electrodomésticos o cualquier otra maquinaria con la intención de reducir la cantidad de desechos electrónicos que acaban en el medio ambiente.* [56]
- Aumentar el sentimiento de inclusión de las personas con deficiencia auditiva ya que alrededor de la mitad de ellas no se sienten del todo incluidas en la sociedad. [57]