UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA APLICACIÓN MÓVIL QUE PERMITA CLASIFICAR SONIDOS PARA PERSONAS CON DISCAPACIDAD AUDITIVA UTILIZANDO DEEP LEARNING.

TELLO MALDONADO ANGEL ANDRES
INGENIERO DE SISTEMAS

MACHALA
2022

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA APLICACIÓN MÓVIL QUE PERMITA CLASIFICAR SONIDOS PARA PERSONAS CON DISCAPACIDAD AUDITIVA UTILIZANDO DEEP LEARNING.

TELLO MALDONADO ANGEL ANDRES
INGENIERO DE SISTEMAS

MACHALA
2022

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

TRABAJO TITULACIÓN
PROPUESTAS TECNOLÓGICAS

DESARROLLO DE UNA APLICACIÓN MÓVIL QUE PERMITA CLASIFICAR SONIDOS PARA PERSONAS CON DISCAPACIDAD AUDITIVA UTILIZANDO DEEP LEARNING.

TELLO MALDONADO ANGEL ANDRES
INGENIERO DE SISTEMAS

RIVAS ASANZA WILMER BRAULIO

MACHALA, 23 DE FEBRERO DE 2022

MACHALA
2022

tesis angel tello

INFORME DE ORIGINALIDAD

8%	7%	0%	6%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Técnica de Machala Trabajo del estudiante	4%
2	repositorio.utmachala.edu.ec Fuente de Internet	2%
3	Submitted to Universidad Internacional de la Rioja Trabajo del estudiante	1%
4	hdl.handle.net Fuente de Internet	<1%
5	Submitted to Esumer Institucion Universitaria Trabajo del estudiante	<1%
6	repositorio.espe.edu.ec Fuente de Internet	<1%
7	moam.info Fuente de Internet	<1%

Excluir citas Activo

Excluir coincidencias < 15 words

CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

El que suscribe, TELLO MALDONADO ANGEL ANDRES, en calidad de autor del siguiente trabajo escrito titulado DESARROLLO DE UNA APLICACIÓN MÓVIL QUE PERMITA CLASIFICAR SONIDOS PARA PERSONAS CON DISCAPACIDAD AUDITIVA UTILIZANDO DEEP LEARNING., otorga a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tiene potestad para otorgar los derechos contenidos en esta licencia.

El autor declara que el contenido que se publicará es de carácter académico y se enmarca en las dispociones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

El autor como garante de la autoría de la obra y en relación a la misma, declara que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asume la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.

Machala, 23 de febrero de 2022

TELLO MALDONADO ANGEL ANDRES
0705297026

Dedicatoria

Dedico primeramente este trabajo a Dios, ya que él es quien guía mi camino todos los días y me llena de fe y esperanza para seguir el camino que elige para mí.

A mi padre Ángel, que con su esfuerzo y perseverancia ha logrado ser un buen hombre y me ha enseñado valores y respeto.

A mi madre Susana, quien con su paciencia y cariño ha sido la base para seguir superándome en cada etapa de mi vida.

A mi hermana Lady, quien me ha demostrado que siempre hay esperanzas por lo más difícil que sea la situación.

A mi familia, mi esposa Karina y mi tesoro Arleth porque son el motor, motivo de mi esfuerzo y amor incondicional.

Finalmente, a los docentes que me ayudaron en todo este proceso de aprendizaje, a mis compañeros con los que compartí y supieron brindar su ayuda en todo este largo camino.

Sr. Angel Andres Tello Maldonado

Agradecimiento

Agradezco primeramente a Dios, por darme la salud, el pan de cada día y la sabiduría en todo este largo camino, en especial agradezco por su ayuda incondicional a mis padres y hermana quienes estuvieron y fueron testigos del esfuerzo.

A mi pequeña familia, porque me brindan su amor, su atención y su comprensión para seguir adelante en cada etapa.

A mi tutor, el Ing. Rivas Asanza Wilmer Braulio, por la confianza y por la guía que me brinda en este proceso de Titulación.

A todos los docentes de esta Universidad, quienes me ayudaron y me formaron en el ámbito profesional.

Resumen

Los teléfonos móviles durante los años han ido evolucionando para brindar a sus usuarios diferentes funcionalidades sin importar cual sea su ámbito o necesidad, volviéndose una herramienta indispensable del día a día, por su capacidad de ofrecer gran utilidad.

Las personas con capacidades reducidas son capaces de comunicarse muchas veces gracias a estos dispositivos, ayudándolos a tareas simples como comunicarse con otros, hasta pedir servicios de emergencia. Todo gracias al dinamismo que ofrecen estos dispositivos.

Con el tiempo se ha visto un creciente interés en la inteligencia artificial, sus usos están limitados a la imaginación de las personas que las ponen en uso, permitiendo a las personas automatizar procesos tediosos y que tomarían mucho tiempo si se hicieran de manera manual.

Ya que los teléfonos móviles han evolucionado a un punto de ser considerados “teléfonos de bolsillo”, se busca crear soluciones que usen inteligencia artificial para lograr resolver los problemas de las personas, sin que tengan que acceder a grandes servidores o a páginas web de manera constante, que todo se pueda lograr con aplicaciones que se puedan instalar en un teléfono.

Con la finalidad de apoyar a las personas por medio de la tecnología, se planea construir una aplicación para teléfonos inteligentes, capaz de reconocer sonidos de alrededor y presentar un resultado de lo que se escucha, usando herramientas de inteligencia artificial, enfocándose en las personas con dificultades auditivas, con el fin de reconocer los sonidos del entorno que no puedan percibir correctamente.

Se realizó este proyecto con herramientas como Android Studio, Teachable Machine, la cual usa como paquete base el algoritmo de Command Speech Recognizer de Tensorflow, el cual posee un flujo de trabajo capaz de identificar palabras aisladas y formar una red de conocimiento a base de estas, funcionando también con sonidos, por lo cual se acopla perfectamente al

proyecto que se está realizando, razón por la cual se buscará exportar en Tensorflow Lite.

Durante la construcción del modelo se busca poner a prueba a la red neuronal con el fin de evaluar su precisión, colocando un índice aceptable para que cumpla con un nivel de satisfacción, lo cual se medirá haciendo uso de las matrices de confusión para cada uno de los dominios que se han definido en el modelo y determinar si el modelo es apto para funcionar en un caso del mundo real.

La forma en la cual funciona la aplicación usando el modelo entrenado, será que por medio del micrófono del dispositivo se pueda escuchar lo que hay en el entorno y pueda entregar un resultado de coincidencia dependiendo de los sonidos que haya reconocido en el entorno y se encuentren en un dominio del modelo.

Gracias a las pruebas que se realizaron en un entorno real, se puede concluir que el modelo cuenta con un alto nivel de aceptación, con una precisión justo a la media necesaria que se propuso para que sea considerado como apto al modelo entrenado, por lo que al modelo se lo considera apto para su aplicación en un caso del mundo real.

Palabras clave: red neuronal, detección de sonidos, Android, Tensorflow, Inteligencia Artificial

Abstract

Over the years, cell phones have been evolving to provide their users with different functionalities no matter what their field or need is, becoming an indispensable day-to-day tool, due to their ability to offer great utility.

People with reduced capacities are able to communicate many times thanks to these devices, helping them with simple tasks such as communicating with others, even asking for emergency services. All thanks to the dynamism offered by these devices.

Over time there has been a growing interest in artificial intelligence, its uses are limited to the imagination of the people who put them to use, allowing people to automate tedious and time-consuming processes that would take too long if done manually.

Since cell phones have evolved to the point of being considered "pocket phones", the idea is to create solutions that use artificial intelligence to solve people's problems, without having to access large servers or web pages constantly, everything can be achieved with applications that can be installed on a phone.

In order to support people through technology, it is planned to build an application for smartphones, capable of recognizing sounds around and present a result of what is heard, using artificial intelligence tools, focusing on people with hearing difficulties, in order to recognize the sounds of the environment that can not perceive correctly.

This project was carried out with tools such as Android Studio, Teachable Machine, which uses as a base package the Command Speech Recognizer algorithm of Tensorflow, which has a workflow capable of identifying isolated words and forming a knowledge network based on these, also working with sounds, so it is perfectly suited to the project being carried out, which is why it will be exported in Tensorflow Lite.

During the construction of the model, the aim is to test the neural network in order to evaluate its accuracy, placing an acceptable index to meet a level of

satisfaction, which will be measured using the confusion matrices for each of the domains that have been defined in the model and determine whether the model is suitable to work in a real-world case.

The way in which the application works using the trained model will be that by means of the device's microphone it will be able to listen to what is in the environment and will be able to deliver a matching result depending on the sounds that have been recognized in the environment and are in a domain of the model.

Thanks to the tests that were performed in a real environment, it can be concluded that the model has a high level of acceptance, with an accuracy just to the necessary average that was proposed to be considered as suitable for the trained model, so the model is considered suitable for application in a real world case.

Keywords: neural network, sound detection, Android, Tensorflow, Artificial Intelligence.

Introducción

La detección de objetos y la inteligencia artificial ha demostrado poseer una gran utilidad en diferentes ámbitos, desde la industria a la medicina, sus usos se limitan a las necesidades que posea el usuario y depende del área en donde se la desee aplicar, pudiendo automatizar procesos que requerían cargas de trabajo humano pesadas.

Las redes neuronales reciben un proceso de aprendizaje o entrenamiento, en el cual suelen ser necesarios equipos de gran poder de procesamiento, pero con la salida de nuevas tecnologías, se puede realizar esto de manera remota, acercando hacia más personas la aplicación de inteligencia artificial.

El entrenamiento de la aplicación en cuestión se lo realiza en la nube gracias a herramientas como Teachable Machine, cuyo núcleo se encuentra en Tensorflow y los algoritmos Speech Command Recognizer.

Para las pruebas se usará un dispositivo móvil capaz con la aplicación instalada, con el cual se podrán realizar pruebas en tiempo real de diferentes sonidos que se van a aplicar en el entorno.

El presente documento consta de la siguiente estructura:

Capítulo 1: se describe la problemática, la justificación y se establecen los requerimientos del prototipo.

Capítulo 2: fundamentación teórica del sistema, diseño del prototipo, ejecución y pruebas del prototipo.

Capítulo 3: se detallan los resultados obtenidos de la aplicación, las conclusiones y recomendaciones

Contenido

Índice de Ilustraciones

Índice de Tablas

1. DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS

1.1 Ámbito de Aplicación: descripción del contexto y hechos de interés

Las personas con dificultades auditivas, en caso de que esta no sea total, dependen mucho del lenguaje de señas en cuando se están comunicando con otras personas, pero quedan totalmente desprevenidos en cuanto a los sonidos provenientes de otros lugares, como lo puede ser el sonido de una calle transitada, animales en la cercanía, etc.

El reconocimiento de estos sonidos es indispensable para ellos ya que estos pueden alertarlos de los peligros que tienen cerca, por lo que se puede poner a disposición de ellos la inteligencia artificial por medio de diferentes medios.

La detección de sonidos requiere de armar un banco de datos con el cual se pueda realizar un entrenamiento y detección de estos sonidos, y esto requiere de una gran cantidad de datos de los cuales se depende la calidad, duración y que no haya ruido blanco u otros sonidos de fondo.

Existen varias herramientas para la creación de una red neuronal capaz de identificar sonidos, pero esta vez se tomó en consideración la herramienta de Teachable Machine con su módulo de reconocimiento de Speech Command Recognizer, el cual puede crear un modelo a través de los diferentes dominios a los que se les asigne los sonidos de la base de datos.

El alcance de este proyecto tiene como finalidad lograr crear un prototipo que realice la detección de diferentes sonidos que se pueden encontrar en el ambiente e identificarlos bajo un dominio definido.

1.2 Establecimiento de requerimientos

La construcción y entrenamiento de la inteligencia artificial capaz de detectar sonidos requiere de algunos procesos antes de lograr un resultado que sea aceptable, por lo que el proyecto se ha planteado de la siguiente manera:

- Recolección de información sobre el estudio de redes neuronales para el entrenamiento y construcción de la misma.
- Construir una base de datos con registros necesarios para el entrenamiento y prueba de la red neuronal, en la cual se va a poseer 7 dominios entre los cuales se encuentran diferentes ruidos que se pueden escuchar de manera cotidiana.
- El entrenamiento de la red neuronal se realizó en la plataforma de Teachable Machine el cual establece el árbol de los dominios creados a partir de las muestras de audio, en el cual se definen los parámetros necesarios para el entrenamiento como lo es: el tamaño por lote, el índice de aprendizaje inicial, las iteraciones por número de épocas.
- Desarrollo de la aplicación móvil con el modelo resultante del entrenamiento de la red neuronal, el cual se encuentra en Tensorflow Lite, que se lo puede acoplar fácilmente en el desarrollo en Android Studio, donde se evaluaran los pesos generados por las detecciones que realiza el modelo al entorno de usuario.
- La fase de pruebas se efectúa por medio del reconocimiento de sonidos en tiempo real.

1.3 Justificación de requerimiento a satisfacer

El proyecto presentado posee su enfoque en el dominio de investigación de Tecnologías, que se asocia con la investigación de Tecnologías de la UTMACH para la producción sustentable.

Las personas con capacidades auditivas reducidas se encuentran en posiciones vulnerables ya que, el oído es el sentido que es capaz de prevenir los peligros que se encuentran alrededor, y dependiendo del nivel de la capacidad auditiva, muchas veces no tiene sentido el uso de un audífono para personas con esta condición.

Por esta razón lo que se busca con esta investigación es acercar la tecnología de la inteligencia artificial y el aprendizaje de máquina a las personas con estas capacidades para ofrecerles más formas de desenvolverse en el día a día. Usando dispositivos que ya se hayan masificado entre la población como lo son los teléfonos inteligentes.

El uso de herramientas como lo es Teachable Machine permite el entrenamiento y la portabilidad de acoplar el modelo a cualquier tipo de interfaz, caracterizándose por el dinamismo que esta plataforma ofrece.

La aplicación móvil se va a desarrollar bajo Android Studio para ofrecer el modelo por medio de la plataforma de teléfonos con el sistema operativo Android.

2. DESARROLLO DEL PROTOTIPO

2.1 Definición del prototipo tecnológico

Las redes convolucionales, son capaces de ofrecer un mayor índice de precisión, la herramienta de Teachable Machine permite el entrenamiento de redes neuronales con altos conjuntos de datos, posicionándose muy bien en el entorno, ya que el algoritmo de reconocimiento de voz lo provee Google.

La arquitectura de la red neuronal se conforma de 2 partes: el entrenamiento del modelo convolucional de predicción, y la fase de exportación del modelo predictivo entrenado.

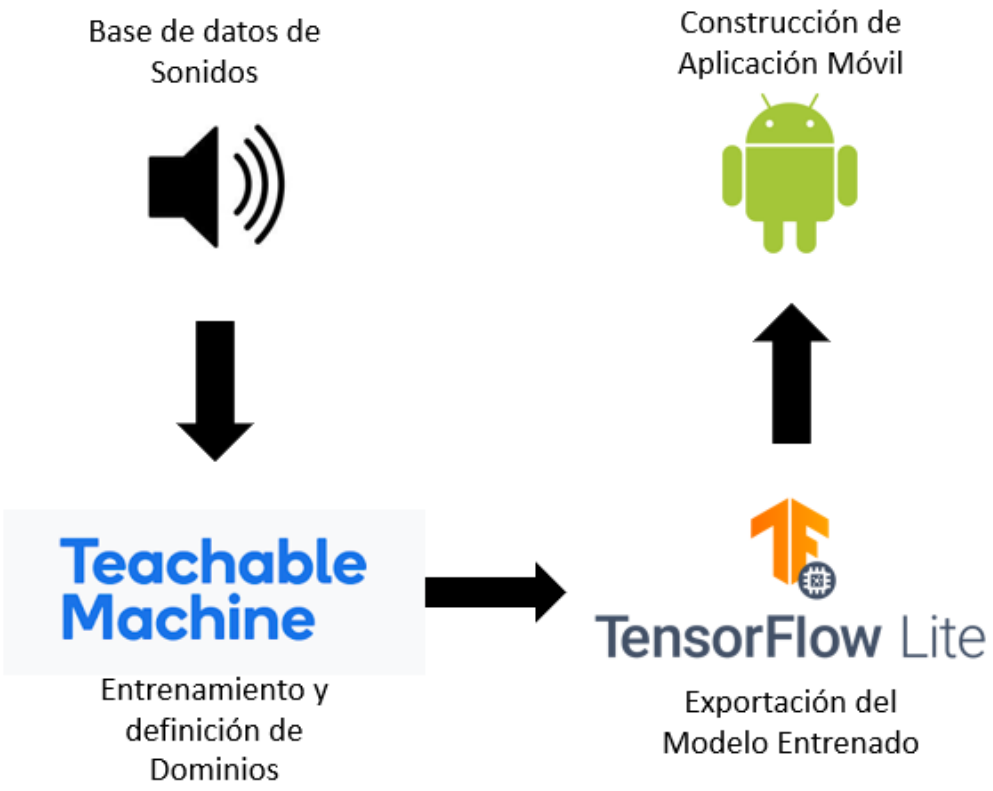

Ilustración 1: Arquitectura de Aplicación
Fuente: Autoría propia

Durante la fase de entrenamiento, a partir de un conjunto de datos se realiza las configuraciones para el entrenamiento de un modelo predictivo en este caso nuestra plataforma de entrenamiento es Teachable Machine; una vez entregue el modelo entrenado se exporta por TensorFlow un paquete, que puede ser usado en cualquier tipo de software o aplicación, en este caso una aplicación

Android gracias a la librería Tensorflow Lite, el cual posee un bajo perfil de consumo de recursos lo que lo hace una herramienta perfecta para la aplicación de redes entrenadas en dispositivos móviles.

2.2 Fundamentación teórica del prototipo

2.2.1 Github

Github ofrece un sitio donde se puede tener respaldo y seguimiento de un código o una base de datos [1]; ofreciendo una plataforma que puede acoplarse fácilmente al trabajo colaborativo y permitiendo la gestión y compartición de código de manera muy fácil. [2]

Es uno de los lideres en el manejo de versionamiento de código, usando la tecnología GIT, se pueden acoplar todo tipo de proyectos a la plataforma sin importar el lenguaje en el que se encuentren desarrollados. [3] [4]

Ofrece muchos módulos y herramientas que dan soporte a los proyectos que se encuentran alojados, como seguimientos de problemas, hilos, cascadas, roles, notificaciones de cambios, configuraciones de seguridad [5] [6], además de facilitar el despliegue y la documentación, y gracias a que tanto la documentación como el código se encuentra en el mismo flujo de trabajo, se pueden asociar rápidamente.

2.2.2 Teachable Machine

Teachable Machine es una herramienta liberada por Google con el fin de facilitar el desarrollo de Inteligencia artificial por medio de la web, se especializa por ser una alternativa fácil y rápida, además de gratuita y accesible para todos. Las capas de dominio que se pueden crear son: imágenes, sonidos, poses o patrones de posición. [7]

Teachable Machine se liberó al público para que el desarrollo de inteligencia artificial no sea un trabajo pesado y para evitar el usar maquinas con un pesado nivel de procesamiento de manera local, por lo que Google, una vez definidos los dominios y parámetros de entrenamiento, realiza el proceso en sus servidores, y entrega el resultado. [8]

2.2.3 Red neuronal convolucional (CNN)

Se conoce como una red neuronal convolucional a un tipo de redes que poseen neuronas artificiales que actúan de manera similar a como lo harían las neuronas humanas de la corteza visual primaria de un cerebro biológico. Basando su tecnología en percepción multicapa, lo cual lo hacen una tecnología preferida para el reconocimiento de imágenes a pesar de que se puede usar para identificar otros tipos de patrones además de los visuales. [9]

Este tipo de redes está formado por varias capas, que se las podría considerar como filtros de varias dimensiones donde se van escogiendo los resultados más apropiados para lograr entregar el resultado de una predicción, este proceso de "filtrado de información" se lo conoce como entrenamiento, lo cual depende tanto de la capacidad de la red como la calidad de los datos, para que el nivel de aceptación de la misma este acorde a las necesidades. [10]

Este tipo de redes son capaces de reconocer diferentes patrones en diferentes tipos de datos, con un proceso de entrenamiento son capaces de identificar los objetos, comportamientos y/o peculiaridades de manera automática al presentar un dato, similar a como lo hiciera una persona, en términos generales, la red aprende a identificar un objeto por medio de visualizar muchas veces el objeto en imágenes de prueba. [11]

2.2.4 Speech Command Recognizer

La clasificación automática es una herramienta que ha evolucionado con el tiempo y de cómo el humano interactúa con los computadores, teniendo muchas aplicaciones tanto en la inteligencia artificial como en la robótica. Con el avance de estas tecnologías se han aplicado diferentes modelos de detección que derivan del aprendizaje profundo, lo cual no es nada factible debido a la energía y el poder de procesamiento que estos exigen. Por medio de diferentes técnicas de reconocimiento, se pueden hacer reconocimientos de diferentes tipos de sonidos, además de lenguajes, a la vez que se les asigna diferentes funciones o resultados a los sonidos que percibe la red. [12]

Para el reconocimiento de voz se han definido redes neuronales convolucionales debido a su manera de procesar y crear una predicción en

base a su proceso de aprendizaje, a pesar de que hay diferentes tipos de aproximamiento, el uso de redes convolucionales es lo que ha dado mejores resultados en diferentes estudios. [13]

Teachable Machine usa el Speech Command Recognizer, lo cual es un módulo incluido en la paquetería de Tensorflow, el cual parte de Web Audio API, que puede realizar inferencias y trasmitir el aprendizaje por medio de aceleraciones de GPU.

El módulo es capaz de identificar palabras insoladas en un clip de audio y crear una base de reconocimiento, de donde parte el aprendizaje, además del "sonido de fondo" o "palabras desconocidas" por defecto.

El flujo de trabajo que sigue el contexto del entrenamiento por audio es el siguiente:

1. Se crea el contexto del audio.
2. Dentro del contexto se crean fuentes.
3. Se crean nodos de efecto, como efectos de reverb, filtros de voz, compresores, etc.
4. Se define la salida final del audio.
5. Se conectan las fuentes a los efectos.

2.2.5 Modelo Entrenado

2.2.5.1 Tensorflow Lite

Tensorflow Lite es el paquete de Tensorflow con la finalidad de funcionar en objetos electrónicos y digitales que no tengan el poder computacional suficiente para ejercer operaciones complejas, siendo usado en dispositivos sensores, computadores vestibles o embebidas y teléfonos móviles. [14]

La peculiaridad de Tensorflow es lograr la exportación de un modelo entrenado, resultado de un proceso de desarrollo para el entrenamiento de una IA, usando el mínimo de recursos para no saturar el dispositivo, usando todas las técnicas necesarias para el reconocimiento de los patrones para lo cual fue creado el modelo. [15]

Tensorflow Lite se puede usar fácilmente en cualquier dispositivo móvil, ya que su funcionamiento de despliegue se acopla a la plataforma donde se lo esté compilando y acoplando, en caso de un Sistema Android, se hace uso de JAVA para las interpretaciones con la API de la red neuronal, y en caso de IOS, se usará Ruby. [16]

2.2.6 Construcción de Aplicación Móvil

2.2.6.1 Sistema Android

Android es un sistema operativo para dispositivos móviles desarrollado por Google con base de Dalvik, que se trata de una variación de Java, actualmente con bases en Kotlin, con el fin ofrecer una colección de paquetes para el acoplamiento e implementación de software y hardware en el sistema móvil. [17]

El sistema operativo Android pertenece a la licencia de código abierto, permitiendo a los desarrolladores móviles tener más facilidad al desplegar aplicaciones para este sistema, en diferencia a su competidor directo, Apple con IOS. [18]

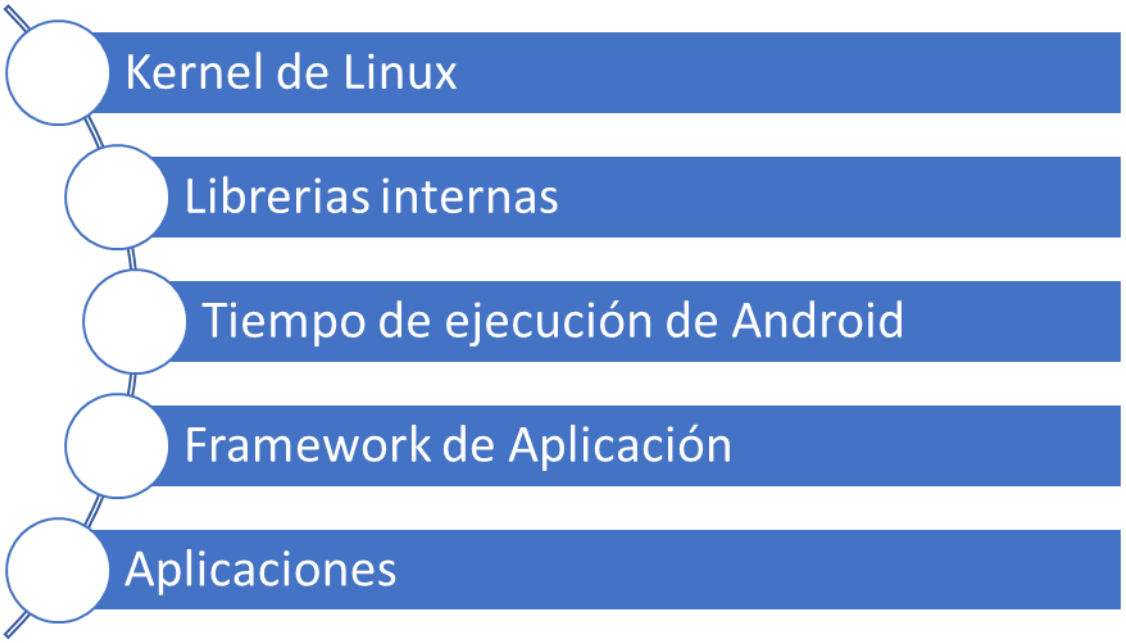

Ilustración 2: Arquitectura de Android SO
Fuente: Autoría propia

2.2.6.2 Android Nougat

Android Nougat, también conocido como Android Versión 6, es la versión que liberó Google en 2016, cuyo proyecto llevaba el nombre de Android Cheesecake. Esta versión de Android recibió mejorar considerables en la interfaz de notificaciones y renderizado de gráficos, además de modos de ahorro de energía para los teléfonos. [19]

Esta versión de Android también optimizó el uso de la pantalla en los teléfonos, ya que permitía la visualización de más de dos de aplicaciones ejecutándose a la vez, lo cual se llamó “ventanas superpuestas” para el uso de dos aplicaciones a la vez en primer plano. [20]

En esta versión también se deja de usar el servicio de apache para la sustentabilidad del sistema operativo y realiza la migración a OpenJDK, con el fin de permitir actualizaciones más fluidas, todos estos cambios basándose en las actualizaciones ejecutadas a segundo plano, permitiendo que el teléfono pueda trabajar o ejecutando procesos aún en segundo plano, tomando en cuenta la experiencia de usuario. [21]

2.2.6.3 Android Studio

Android Studio se trata de un entorno de desarrollo oficializado por Google, para la construcción, depuración y versionamiento de aplicaciones Android, basado en la base de IntelliJ, este IDE se ha desarrollado con la finalidad de ofrecer las novedades de desarrollo para los sistemas operativos Android más recientes. [22]

Este sistema permite la integración con Gradle, SDKS de Android, paquetería de Google, Firebase, Cloud, tecnología Git, además de permitir el desarrollo de interfaz graficas por medio de XML con etiquetas de HTML para un manejo más sencillo a la hora de diseñar mejores interfaces. [23]

2.2.6.4 Android SDK

Android SDK, por sus siglas “Software Development Kit”, es el conjunto de herramientas que permite el desarrollo de aplicaciones para sistemas Android. Los SDK poseen versiones y adaptabilidades para el tipo o versión del sistema operativo para el cual se esté desarrollando, por lo cual los proyectos de desarrollo móviles siempre poseen un SO objetivo. [24]

Los paquetes SDK son una suite completa para el desarrollo, en este caso Android, posee la paquetería Gradle para compilación y pruebas, Java como lenguaje de procesamiento base, XHTML como administrador de interfaces gráficas, y cualquier paquetería adicional de Google que se desee usar, además de poder agregarle paqueterías externas. [25]

2.3 Objetivos del prototipo

2.3.1. Objetivo general

Desarrollar una aplicación móvil como herramienta de soporte para personas con dificultades auditivas que permita reconocer diferentes sonidos del entorno.

2.3.2. Objetivos específicos

- Investigar sobre el diseño y entrenamiento de redes neuronales convolucionales.
- Entrenar una red neuronal con datos obtenidos de repositorios de internet.
- Evaluar los modelos obtenidos validándolos con artefactos y métodos aplicables a la inteligencia artificial.
- Desarrollar una aplicación móvil que permita realizar casos de uso reales en el entorno.

2.4 Diseño del prototipo

Para la creación del prototipo, se ha tomado en cuenta 3 fases de desarrollo;

- La primera fase es la de entrenamiento del modelo, con los datos obtenidos de repositorios de Kaggle y YouTube se armará un entrenamiento por medio de Teachable Machine para la creación de un modelo de detección de sonidos.
- La segunda fase abarca lo que son las pruebas de validación del modelo, en el cual se busca obtener un grado de aceptación suficiente para considerar el modelo efectivo, y pasar a la siguiente fase.
- La tercera fase es la construcción de la aplicación móvil para la ejecución del modelo entrenado aceptado, para su aplicación en un entorno real.

2.5 Ejecución y ensamblaje del prototipo

2.5.1 Desarrollo del Modelo

Como se mencionaba anteriormente, el modelo se encuentra construido en Teachable Machine, en el cual se subirán las muestras de audio anteriormente seleccionadas para desarrollar el modelo.

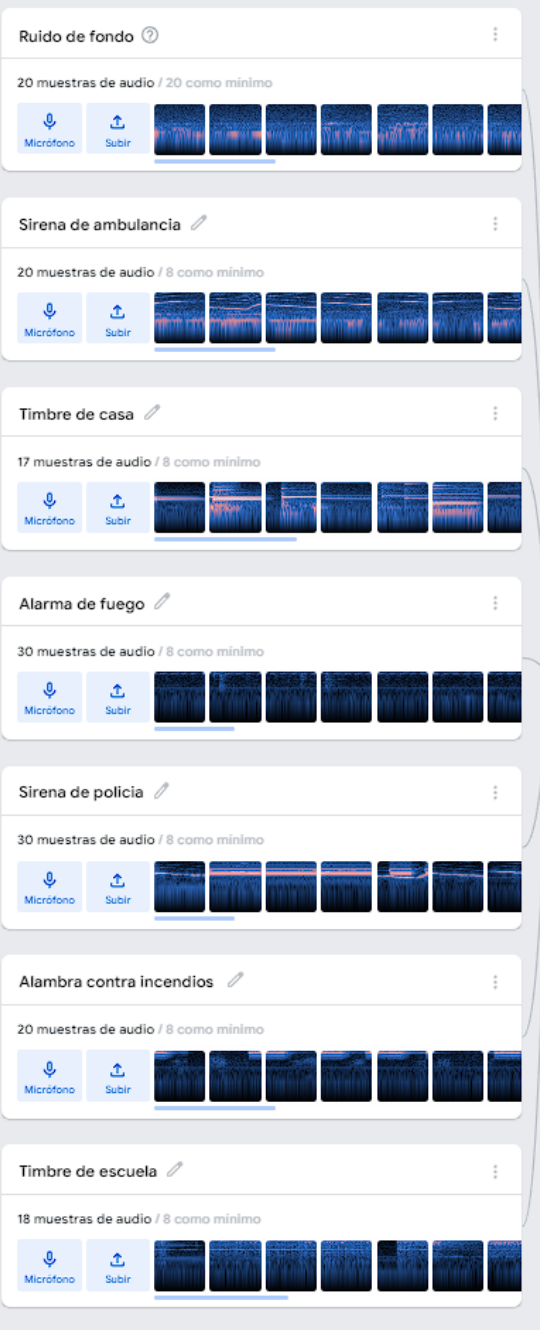

Ilustración 3: Dominios del modelo
Fuente: Autoría propia

En este modelo se deben definir los parámetros de entrenamiento para el desarrollo del mismo, en este caso los parámetros de entrenamiento, es decir, un entrenamiento de 50 épocas.

Una vez realizado el modelo se puede ver una vista previa del mismo.

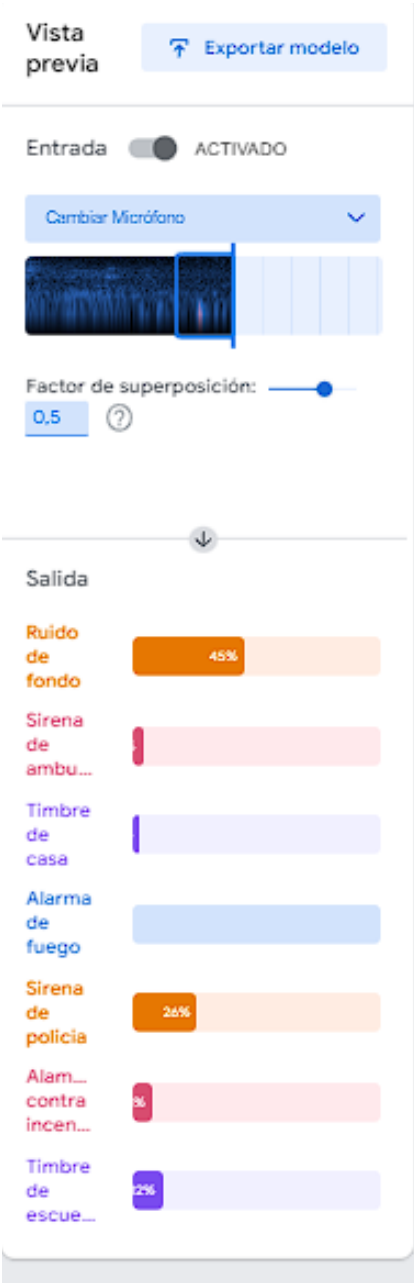

Ilustración 4: Vista previa del modelo
Fuente: Autoría propia

Una vez realizado el modelo Teachable Machine ofrece la opción de poder exportar el modelo para su respectivo uso en una aplicación, como se puede ver a continuación.

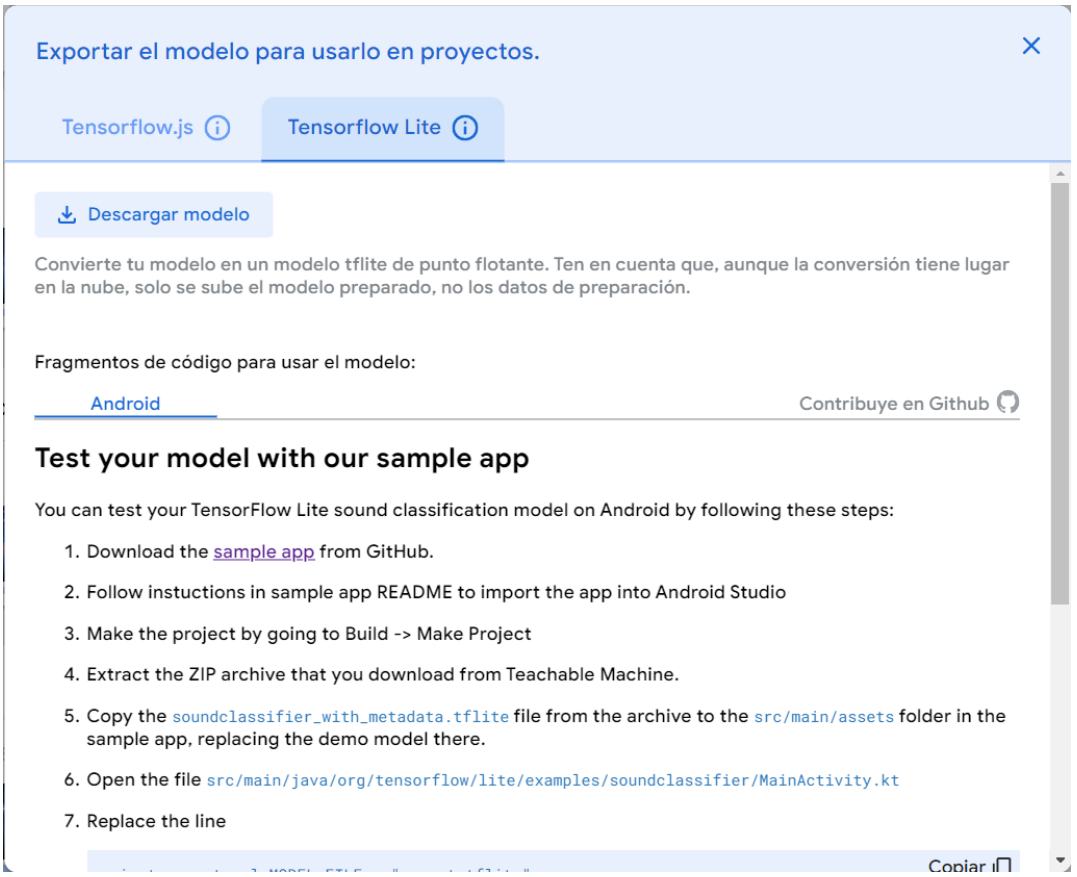

Ilustración 5: Exportación del modelo por TFLite
Fuente: Autoría propia

Ya que se está desarrollando una aplicación móvil, se va a exportar el modelo por TensorFlow Lite, lo cual no permite descargarlo de manera directa a la computadora para hacer uso del mismo.

2.5.2 Desarrollo de la aplicación

El prototipo se desarrolló en Android Studio, usando components y screens para las pantallas de la aplicación, y así mismo usando los permisos del sistema operativo para el uso del micrófono y el envió de notificaciones, y también los permisos de ejecución en segundo plano.

La distribución del proyecto se clasifica en los recursos de drawable, los recursos de layout, los recursos de mipmap, y los Activity para la llamada de funciones y hacer uso del modelo entre MainActivity y ProbabilitesAdapter.

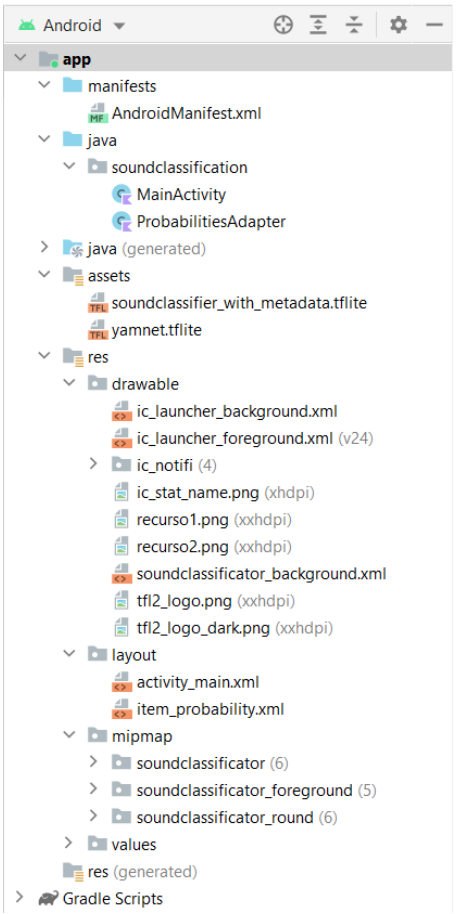

Ilustración 6: Distribución del proyecto en Android Studio
Fuente: Autoría propia

2.5.2.1 MainActivity

El archivo MainActivity realiza las importaciones necesarias para el manejo de notificaciones, administración y uso de paquetes agregados, escucha y grabación de sonido, manejo de pantallas, notificaciones, vibración del

dispositivo, además de paquetes importados desde los archivos de TensorFlow Lite.

```
import android.Manifest
import android.annotation.SuppressLint
import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.Context
import android.content.pm.PackageManager
import android.graphics.Color
import android.media.AudioRecord
import android.os.*
import android.util.Log
import android.view.WindowManager
import androidx.annotation.RequiresApi
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat
import androidx.core.content.ContextCompat
import androidx.core.os.HandlerCompat
import org.tensorflow.lite.examples.soundclassifier.databinding.ActivityMainBinding
import org.tensorflow.lite.task.audio.classifier.AudioClassifier
import android.os.Build
import android.os.Vibrator
import org.tensorflow.lite.examples.soundclassifier.R.drawable
```

Ilustración 7: Importaciones del MainActivity
Fuente: Autoría propia

Las variables a continuación son necesarias para obtener el porcentaje de probabilidades, la clasificación de audio, la grabación de audio, y el intervalo entre los dominios, además del handler para el funcionamiento en segundo plano.

```
class MainActivity : AppCompatActivity() {

  private val probabilitiesAdapter by lazy { ProbabilitiesAdapter() }
  private var audioClassifier: AudioClassifier? = null
  private var audioRecord: AudioRecord? = null
  private var classificationInterval = 500L // Con qué frecuencia debe ejecutarse la clasificación en milisegundos
  private lateinit var handler: Handler // Controlador de subprocesos en segundo plano para ejecutar la clasificación
```

Ilustración 8: Variables privadas de MainActivity
Fuente: Autoría propia

El código a continuación permite encender el interruptor para que se active la grabación por el micrófono del dispositivo y empezar la detección del clasificador de sonidos.

```
    // Interruptor de entrada para activar/desactivar la clasificación
    keepScreenOn(inputSwitch.isChecked)
    inputSwitch.setOnCheckedChangeListener { _, isChecked ->
      if (isChecked) startAudioClassification() else stopAudioClassification()
      keepScreenOn(isChecked)
    }
  }
```

Ilustración 9: Código del interruptor de encendido

Fuente: Autoría propia

El handler se crea para poder definir un hilo de subproceso que permita a la aplicación poder trabajar minimizada y en segundo plano, esto permite que se siga ejecutando a pesar de que no tenga el enfoque o que el teléfono se encuentre en modo de bloqueo.

```
64    // Crear un controlador para ejecutar la clasificación en un subproceso de fondo
65    val handlerThread = HandlerThread( name: "backgroundThread")
66    handlerThread.start()
67    handler = HandlerCompat.createAsync(handlerThread.looper)
```

Ilustración 10: Handler del hilo de proceso
Fuente: Autoría propia

Este código permite que la aplicación pueda seguir trabajando y retomar la ejecución del algoritmo si se realiza un cambio de ventanas o se activa la funcionalidad multiventana.

```
171  override fun onTopResumedActivityChanged(isTopResumedActivity: Boolean) {
172      // Maneja el evento "superior" reanudado en un entorno de múltiples ventanas
173      if (isTopResumedActivity) {
174          startAudioClassification()
175      } else {
176          startAudioClassification()
177      }
178  }
```

Ilustración 11: Código - Resumir actividad
Fuente: Autoría propia

Se crean variables privadas con el nombre de los canales de notificación para que la aplicación pueda enviar y mostrar correctamente los resultados en pop ups creados desde el sistema operativo.

```
38    // Variables para la notificacion
39    private val channelID = "channelID"
40    private val channelName = "sound classification"
41    private val notificationId = 0
```

Ilustración 12: Variables privadas de notificaciones
Fuente: Autoría propia

En el código se define la prioridad de la notificación en alto para que siempre se muestra arriba del todo en la pantalla, además de aplicarle estilos al canal de notificaciones, que en este caso va a ser de color rojo, va a activar los leds

de notificaciones del teléfono en caso de que los posea, y va a activar la vibración.

```
69    // Permisos de Notificacion
70    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
71      val importance = NotificationManager.IMPORTANCE_HIGH
72      val channel = NotificationChannel(channelID,channelName,importance).apply {  this: NotificationChannel
73        lightColor = Color.RED
74        enableLights( lights: true)
75        enableVibration( vibration: true)
76      }
77      val manager = getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
78      manager.createNotificationChannel(channel)
79    }
```

Ilustración 13: Objeto constructor de notificación
Fuente: Autoría propia

El siguiente código se ejecuta para poder hacer uso del micrófono, enviando una función RequestMicrophone, en donde se envía el contexto del objeto indicando que se desea grabar un audio, se coloca dentro de un if para que se pida la autorización del usuario en caso de la aplicación no posea los permisos de uso de micrófono cuando se ejecute.

```
196   // Pregunta al usuario sobre el uso del microfono
197   @RequiresApi(Build.VERSION_CODES.M)
198   private fun requestMicrophonePermission() {
199     if (ContextCompat.checkSelfPermission(
200                     context: this,
201                     Manifest.permission.RECORD_AUDIO
202           ) == PackageManager.PERMISSION_GRANTED
203     ) {
204       startAudioClassification()
205     } else {
206       requestPermissions(arrayOf(Manifest.permission.RECORD_AUDIO), REQUEST_RECORD_AUDIO)
207     }
208   }
```

Ilustración 14: Petición de permisos de micrófono
Fuente: Autoría propia

Una vez se haga la petición del permiso para el uso del micrófono del dispositivo, se realiza un log para saber si se otorgó correctamente el mismo o si no se ejecutó el procedimiento.

```
187    // Obtiene la respuesta del usuario sobre el uso del microfono
188    override fun onRequestPermissionsResult(
189            requestCode: Int,
190            permissions: Array<out String>,
191            grantResults: IntArray
192    ) {
193      if (requestCode == REQUEST_RECORD_AUDIO) {
194        if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
195          Log.i(TAG,  msg: "Permiso Concedido :)")
196          startAudioClassification()
197        } else {
198          Log.e(TAG,  msg: "Permiso Denegado :(")
199        }
200      }
201    }
```

Ilustración 15: Resultado de petición de permisos
Fuente: Autoría propia

El algoritmo de TensorFlow para la clasificación de la entrada de audio se ejecuta solo si el permiso del micrófono se encuentra otorgado.

```
86      // Si el permiso de microfono esta habilitada se ejecuta la classificacion
87      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
88        requestMicrophonePermission()
89      }else{
90        startAudioClassification()
91      }
92    }
```

Ilustración 16: Confirmación de permisos de micrófono
Fuente: Autoría propia

La función KeepScreenOn permite que la pantalla del dispositivo no entre en reposo cuando se esta usando la aplicación, esto usando flags del SDK de Android y la dependencia window.

```
218    // Permite que nunca se apague la pantalla mientras se está en la aplicacion
219    private fun keepScreenOn(enable: Boolean) =
220      if (enable) {
221        window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)
222      } else {
223        window.clearFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)
224      }
```

Ilustración 17: Código - Mantener pantalla activa
Fuente: Autoría propia

La clase de MainActivity posee variables constantes que se usan a lo largo de la ejecución del software, REQUEST_RECORD_AUDIO indica el valor del codec del audio usado para la detección, TAG indica el tipo de información se que va a tomar, MODEL_FILE indica el nombre del archivo que se está usando

para tomar las métricas de la clasificación del sonido y MINIMUM_DISPLAY_THRESHOLD indica el intervalo en el que se realiza la nueva toma de la muestra de audio.

```
226     companion object {
227         const val REQUEST_RECORD_AUDIO = 1337
228         private const val TAG = "AudioDemo"
229         private const val MODEL_FILE = "soundclassifier_with_metadata.tflite" // El modelo ya entrenado en teachable machine
230         private const val MINIMUM_DISPLAY_THRESHOLD: Float = 0.3f
231     }
232 }
```

Ilustración 18: Variables globales de MainActivity
Fuente: Autoría propia

Para que se pueda ejecutar la clasificación del audio, se debe de crear una variable que guarde el estado del clasificador, en este caso se debe de comprobar que la variable audioClassifier no se envíe como null para evitar el surgimiento de errores, en tal caso, se inicia el clasificador creándose a partir del modelo entrando y cargado por TensorFlow Lite (classifier), luego de esto se le define una entrada de audio para realizar la predicción (audioTensor), y al final la variable donde se resguarda el audio tomado por el dispositivo (record); así al final empezando la toma de la muestra de audio.

```
91  private fun startAudioClassification() {
92      // Si el clasificador de audio está inicializado y en ejecución.
93      if (audioClassifier != null) return;
94
95      // Inicializar el clasificador de audio
96      val classifier = AudioClassifier.createFromFile( context: this, MODEL_FILE)
97      val audioTensor = classifier.createInputTensorAudio()
98
99      //Inicializar la grabadora de audio
100     val record = classifier.createAudioRecord()
101     record.startRecording()
```

Ilustración 19: Ejecución de la clasificación de audio
Fuente: Autoría propia

Cuando se tienen los permisos y se han definido las variables se empieza con la ejecución del código que toma el audio de Tensor y ejecuta la clasificación, en el momento que se activa el switch, una vez ejecutada la clasificación, el modelo entrega el resultado en un objeto, donde se los ordena de forma descendente en base al porcentaje de coincidencia, luego de esto, se imprime en pantalla el porcentaje que haya llegado a más de 90% de coincidencia de entre todos los dominios que se tiene en la pantalla, y a su vez se define la

notificación y se la envía al canal de esta, para luego ejecutar la vibración del dispositivo.

```
//Se realiza la ejecucion de la clasificacion
val run = object : Runnable {
  @SuppressLint( ...value: "ServiceCast")
  override fun run() {
    val startTime = System.currentTimeMillis()
    // Cargue la última muestra de audio
    audioTensor.load(record)
    val output = classifier.classify(audioTensor)
    // Filtre los resultados por encima de un determinado umbral y ordénelos de forma descendente
    val filteredModelOutput = output[0].categories.filter { it: Category!
      it.score > MINIMUM_DISPLAY_THRESHOLD
    }.sortedBy { it: Category!
      -it.score
    }
    // Filtre los puntajes mayores a 90 y muestrelo en una notificacion, ademas haga una pequeña vibracion de 750 milisegundos
    val text = output[0].categories.filter { it.score > 0.90}.toString()
    if(text != "[]") {
      val textf = text.substring(text.indexOf( string: "Category")+12,text.indexOf( string: "displayName")-3)
      if(textf != "Ruido de fondo"){
        val notification = NotificationCompat.Builder(applicationContext, channelID).also { it: NotificationCompat.Builder
          it.setContentTitle("Sound Classification")
          it.setContentText("Se detecto un Sonido de " + textf)
          it.setSmallIcon(drawable.ic_notifi)
          it.setPriority(NotificationCompat.PRIORITY_MAX)
          it.setAutoCancel(false)
        }.build()
        val notificationManager = NotificationManagerCompat.from(applicationContext)
        notificationManager.notify(notificationId,notification)
        val vibrator = getSystemService(Context.VIBRATOR_SERVICE) as Vibrator
        vibrator.vibrate( milliseconds: 750)
      }
    }
```

Ilustración 20: Predicción y muestra de resultados
Fuente: Autoría propia

Se tiene un código para detener la clasificación y toma de muestras una vez se desactive el switch, lo cual es colocar todas las variables de detección en null.

```
private fun stopAudioClassification() {
    handler.removeCallbacksAndMessages( token: null)
    audioRecord?.stop()
    audioRecord = null
    audioClassifier = null
}
```

Ilustración 21: Detener clasificación de audio
Fuente: Autoría propia

2.5.2.2 Clase ProbabilitiesAdapter

La clase de ProbilitiesAdapter es donde se realizan todos los cálculos de predicción y muestreo de resultados que se dan una vez se coloca la muestra en el modelo de TensorFlow entrenado, en donde también se cargar los

recursos visuales, pudiendo mostrar por barras de progreso el índice de coincidencia de todos los dominios que se tomaron en cuenta.

```
30    class ViewHolder(private val binding: ItemProbabilityBinding) :
31      RecyclerView.ViewHolder(binding.root) {
32      fun bind(position: Int, label: String, score: Float) {
33        with(binding) { this: ItemProbabilityBinding
34          labelTextView.text = label
35          progressBar.progressBackgroundTintList = progressColorPairList[position % 3].first
36          progressBar.progressTintList = progressColorPairList[position % 3].second
37
38          val newValue = (score * 100).toInt()
39          // Crea la animacion de la barra de proceso de prediccion
40          val animation =
41            ObjectAnimator.ofInt(progressBar, propertyName: "progress", progressBar.progress, newValue)
42          animation.duration = 100
43          animation.interpolator = AccelerateDecelerateInterpolator()
44          animation.start()
45        }
46      }
47
48      companion object {
49        // Color de fondo y Color de progreso
50        private val progressColorPairList = listOf(
51          ColorStateList.valueOf(0xffCEE7FF.toInt()) to ColorStateList.valueOf(0xff00BFFF.toInt()),
52          ColorStateList.valueOf(0xfff7e3e8.toInt()) to ColorStateList.valueOf(0xffc95670.toInt()),
53          ColorStateList.valueOf(0xffecf0f9.toInt()) to ColorStateList.valueOf(0xff714Fe7.toInt()),
54        )
55      }
56    }
57  }
```

Ilustración 22: Clase ProbabilitiesAdapter
Fuente: Autoría propia

2.5.2.3 Android Manifest

El Manifest de un proyecto de Android es donde se definen las configuraciones iniciales para el uso de paquetería o servicios que se van a necesitar posteriormente en la ejecución del programa, es un archivo que el SDK de Android genera, en este caso se le ha definido un nombre, icono, etiqueta y tema, además del nombre del paquete que se usa para la importación del modelo entrenado de TensorFlow, y los permisos de vibración y grabación de audio.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="org.tensorflow.lite.examples.soundclassifier">

    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.VIBRATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/soundclassificator"
        android:label="Sound Classification"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name="org.tensorflow.lite.examples.soundclassifier.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Ilustración 23: Android Manifest
Fuente: Autoría propia

2.5.3. Ejecución del aplicativo móvil

Una vez se instale el programa en un dispositivo Android, se podrá ver el icono asignado además del nombre de la aplicación, pudiéndose crear un acceso directo de la misma.

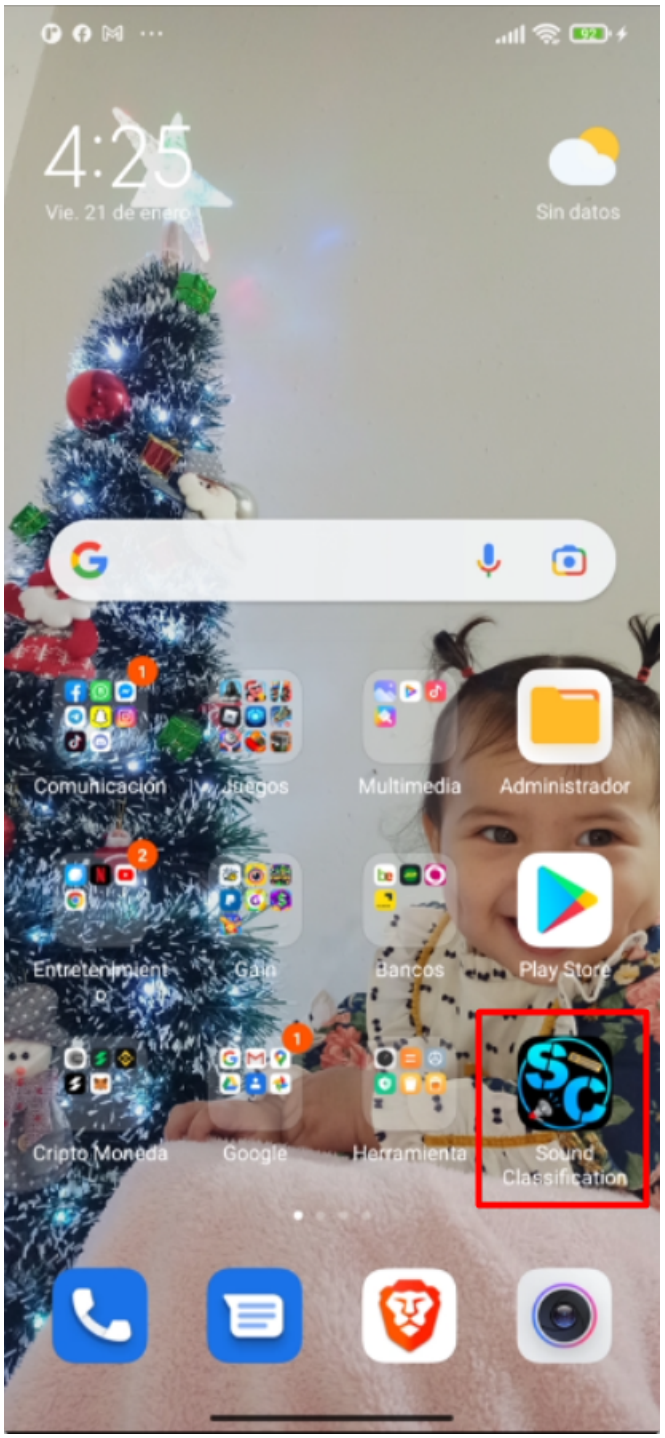

Ilustración 24: Ícono y título de aplicación
Fuente: Autoría propia

La pantalla inicial de la aplicación muestra el switch que permite encender el micrófono para la toma de las muestras de audio, además de una barra que indica el número del dominio además del valor del porcentaje que se tiene.

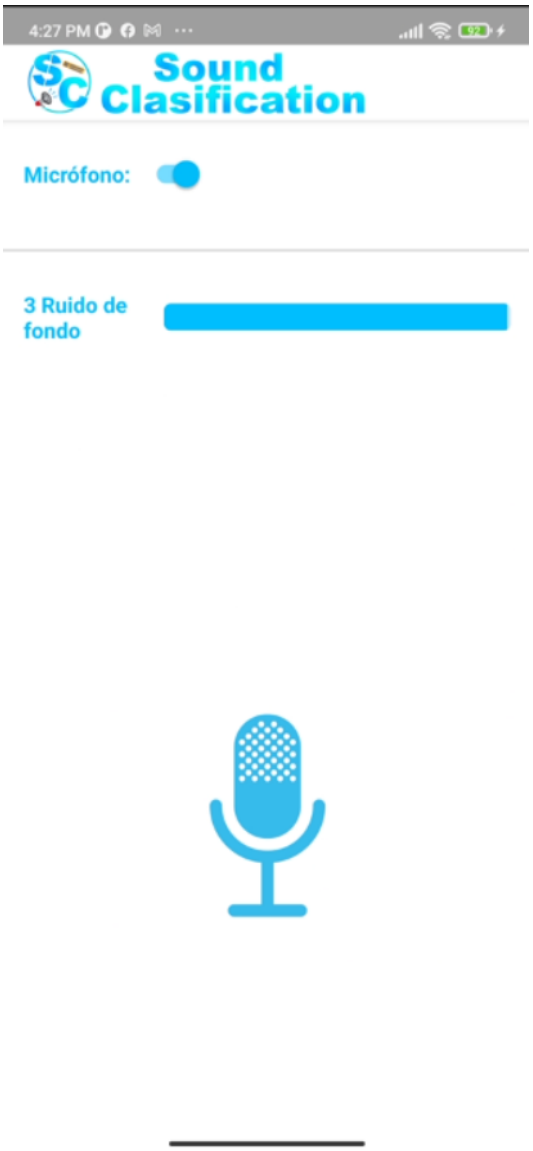

Ilustración 25: Pantalla principal
Fuente: Autoría propia

Para poder hacer uso del micrófono, la primera vez que se ejecute, y cada vez que la aplicación no presente los permisos necesarios, aparecerá la pantalla haciendo petición de los mismos para la ejecución del algoritmo

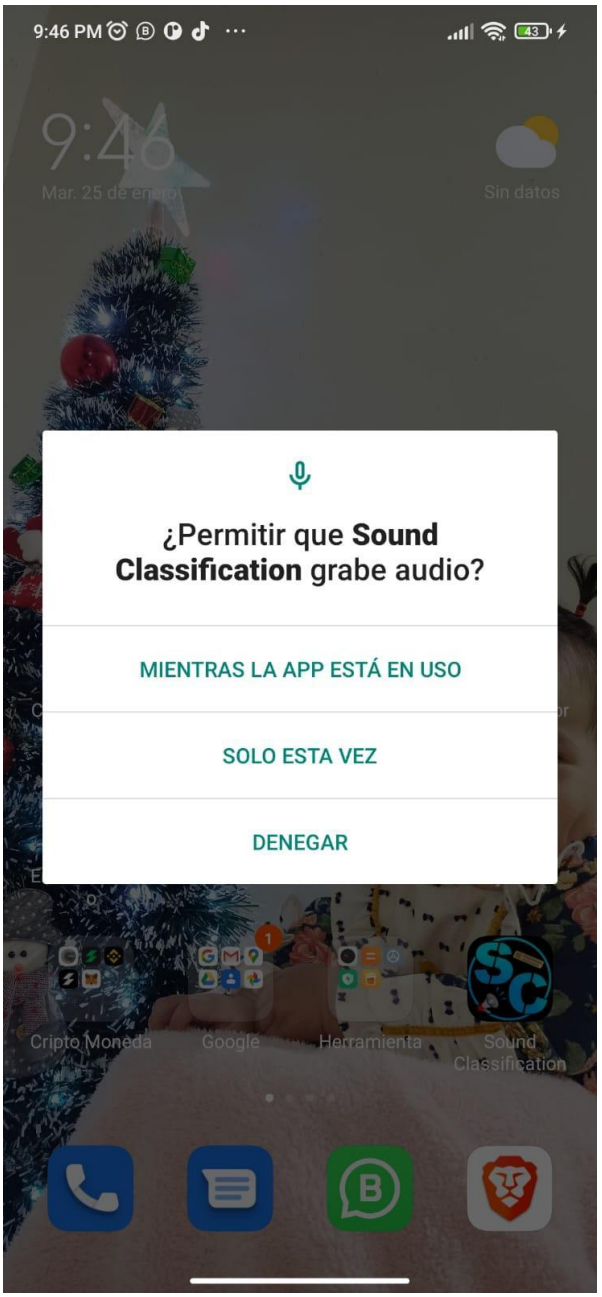

Ilustración 26: Pantalla de petición de permisos
Fuente: Autoría propia

En caso de que el micrófono no se encuentre activado con el switch, se mostrará un icono indicando, en el centro de la pantalla.

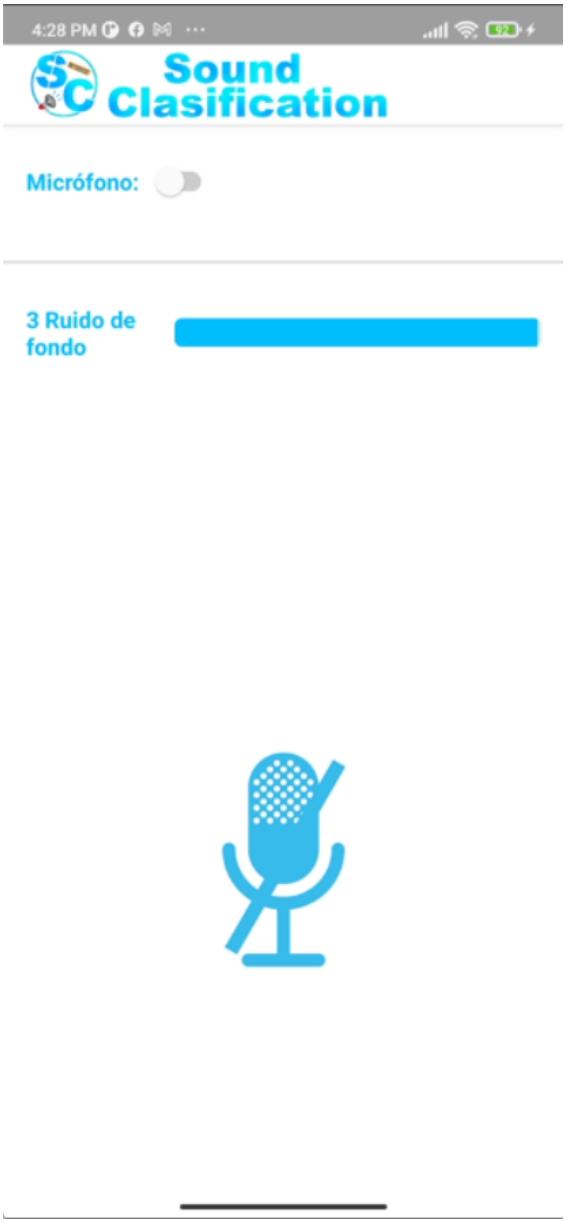

Ilustración 27: Pantalla principal con micrófono desactivado
Fuente: Autoría propia

Una vez se tome una muestra y se haga la evaluación, se muestran las coincidencias en la pantalla de la aplicación, y se envía una notificación vibración indicando el porcentaje de coincidencia más alto.

Ilustración 28: Resultado de detección de sonido
Fuente: Autoría propia

La aplicación también puede seguir funcionando en segundo plano, por lo que en caso de que siga detectando sonidos que se encuentren en el modelo entrenado, seguirá mostrando una notificación con la predicción que posea el índice mayor.

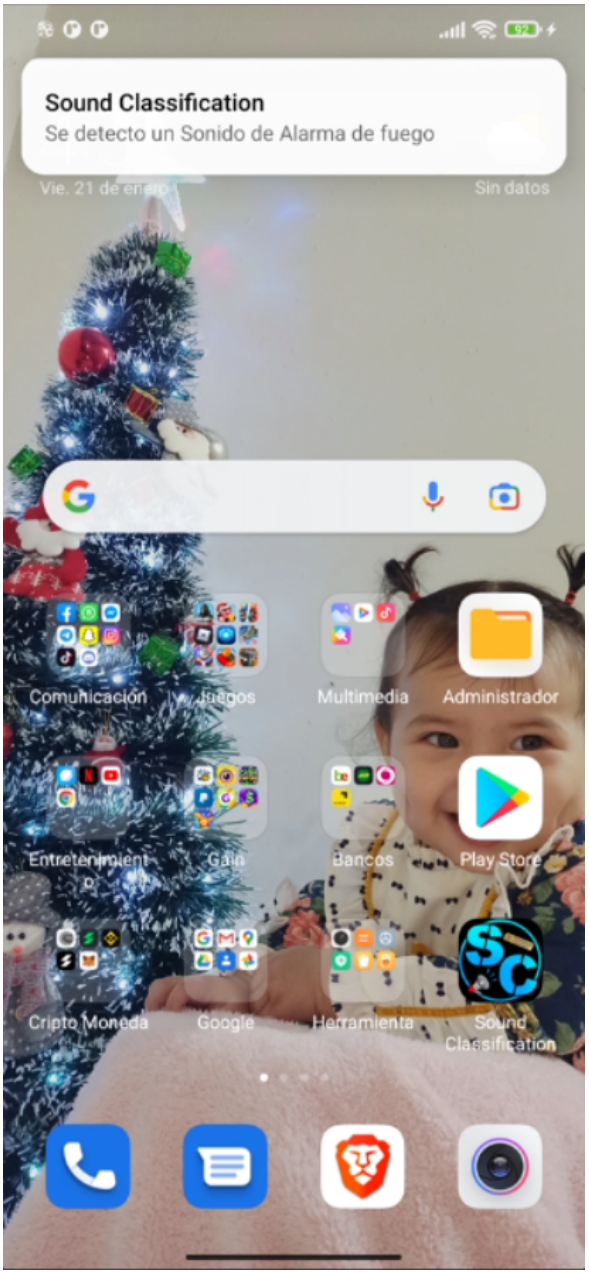

Ilustración 29: Detección de sonido en segundo plano con notificación
Fuente: Autoría propia

Ya que el canal de la notificación está definida como prioridad alta, la notificación de la aplicación siempre se mostrará arriba, a menos que haya notificaciones del sistema anteriormente.

Ilustración 30: Notificación de predicción en pantalla de bloqueo
Fuente: Autoría propia

3. EVALUACIÓN DEL PROTOTIPO

3.1 Plan de evaluación

Los resultados obtenidos se sometieron a pruebas y análisis comparativos para formar un índice de precisión y predicción, en las cuales se hacen pruebas de error con el fin de verificar la efectividad de un modelo de predicción.

Para esta etapa se debe de implementar “Ground Truth” o verdad fundamental, que se focaliza en datos que son meramente observados y sé puede clasificar a simple vista. [26] [27]

3.2 Pruebas de validación del modelo

3.2.1 Matriz de Confusión

Una matriz de confusión es un método que se usa en el análisis de modelos de detección para medir un índice calculando la cantidad de resultados correctos que pueda obtener el algoritmo, lo cual se lo considera como “Verdaderos Positivos”, los resultados que fueron considerados verdaderos de manera errónea llamados “Falsos Positivos”, los resultados que fueron correctamente detectados como negativos llamados “Verdaderos Negativos”, y los resultados verdaderos detectados como falsos, que se los llama “Falsos Negativos”. [28] [29]

Para un mejor manejo de estos conceptos, se les colocará una abreviatura:

Resultados	**Abreviatura**
VP	Verdadero Positivo
VN	Verdadero Negativo
FN	Falso Negativo
FP	Falso Positivo

Tabla 1: Abreviatura de Matriz de Confusión
Fuente: Autoría Propia

Para cada dominio se va a aplicar la siguiente tabla de conteo para los resultados, ya que, para ponderar índices de varianza y precisión del modelo en su totalidad, se requiere de aplicar una matriz de confusión para todos los dominios presentes en esta.

La siguiente tabla es solo un modelo para aplicar a todos los dominios:

		Modelo Predictivo	
		Positivos	**Negativos**
Mundo Real	**Positivos**	Verdadero Positivo (VP)	Falso Negativo (FN)
	Negativos	Falso Positivo (FP)	Verdadero Negativo (VN)

Tabla 2: Tabla de Conteo de Matriz de Confusión
Fuente: Autoría Propia

Con los valores que se forman en la matriz de confusión, se puede aplicar fórmulas para calcular exactitud, precisión, sensibilidad, puntos de referencia, etc. [30]

F-Score

Es un índice formado entre la relación de la precisión con sensibilidad, sirve como referencia entre las relaciones de los resultados positivos del modelo, el cual se calcula multiplicando por 2 el cociente del producto entre la precisión por la sensibilidad para la suma entre la precisión y la sensibilidad.

$$F - Score = 2 * \frac{Precisión * Sensibilidad}{Precisión + Sensibilidad}$$

Sensibilidad

Se entiende por la sensibilidad a la efectividad que posee un modelo para la detección de valores positivos, el índice se pondera por la relación que resulta del cociente entre los verdaderos positivos para la suma de los verdaderos positivos más los falsos negativos

$$Sensibilidad = \frac{VP}{VP + FN}$$

Precisión

La precisión se entiende como el porcentaje de exactitud de los resultados de la detección o predicción, el índice se forma por la relación de cociente entre los verdaderos positivos para la suma de los verdaderos positivos más los falsos negativos.

$$Precisión = \frac{VP}{VP + FP}$$

Exactitud

La exactitud es un porcentaje que indica cual es el índice de resultados se predijeron correctamente.

$$Exactitud = \frac{VP + VN}{VP + FP + VN + FN}$$

3.3 Resultados de la evaluación

La prueba se la realizó colocando clips de audio del mundo real en la aplicación, llenando las tablas de los dominios existentes en el modelo. A cada dominio se le aplica la prueba de matriz de confusión para extraer los índices de análisis para cada uno de estos, para cada dominio se han tomado 100 muestras de sonido aleatorias que no fueron usadas durante el entrenamiento del modelo.

RUIDO DE FONDO

		Modelo Predictivo	
		Positivos	**Negativos**
Mundo Real	**Positivos**	41	15
	Negativos	14	30
TOTAL	100		

Tabla 3: Matriz de Confusión - Ruido de Fondo
Fuente: Autoría Propia

SIRENA DE AMBULANCIA

		Modelo Predictivo	
		Positivos	**Negativos**
Mundo Real	**Positivos**	57	17
	Negativos	13	13

Tabla 4: Matriz de Confusión - Sirena de Ambulancia
Fuente: Autoría Propia

SIRENA DE POLICIA

		Modelo Predictivo	
		Positivos	**Negativos**
Mundo Real	**Positivos**	49	10
	Negativos	17	24

Tabla 5: Matriz de Confusión - Sirena de Policía
Fuente: Autoría Propia

ALARMA DE FUEGO

		Modelo Predictivo	
		Positivos	**Negativos**
Mundo Real	**Positivos**	40	21
	Negativos	7	32

Tabla 6: Matriz de Confusión – Alarma de Fuego

Fuente: Autoría Propia

TIMBRE DE CASA

		Modelo Predictivo	
		Positivos	**Negativos**
Mundo Real	**Positivos**	43	19
	Negativos	11	27

Tabla 7: Matriz de Confusión – Timbre de Casa
Fuente: Autoría Propia

ALARMA CONTRA INCENDIOS

		Modelo Predictivo	
		Positivos	**Negativos**
Mundo Real	**Positivos**	46	14
	Negativos	16	24

Tabla 8: Matriz de Confusión - Alarma contra Incendios
Fuente: Autoría Propia

TIMBRE DE ESCUELA

		Modelo Predictivo	
		Positivos	**Negativos**
Mundo Real	**Positivos**	51	18
	Negativos	12	19

Tabla 9: Matriz de Confusión - Timbre de Escuela
Fuente: Autoría Propia

Luego de hacer el análisis por medio de matriz de confusión se procede a realizar el análisis de los índices de precisión, exactitud, sensibilidad y F-Score.

RUIDO DE FONDO

Exactitud	0,71
Precisión	0,74545455
Sensibilidad	0,73214286
F-Score	0,73873874

Tabla 10: Índices - Ruido de Fondo
Fuente: Autoría Propia

SIRENA DE AMBULANCIA

Exactitud	0,7
Precisión	0,81428571
Sensibilidad	0,77027027
F-Score	0,79166667

Tabla 11: Índices - Sirena de Ambulancia
Fuente: Autoría Propia

SIRENA DE POLICIA

Exactitud	0,73
Precisión	0,74242424
Sensibilidad	0,83050847
F-Score	0,784

Tabla 12: Índices - Sirena de Policía
Fuente: Autoría Propia

ALARMA DE FUEGO

Exactitud	0,72
Precisión	0,85106383
Sensibilidad	0,6557377
F-Score	0,74074074

Tabla 13: Índices - Alarma de Fuego
Fuente: Autoría Propia

TIMBRE DE CASA

Exactitud	0,7
Precisión	0,7962963
Sensibilidad	0,69354839
F-Score	0,74137931

Tabla 14: Índices - Timbre de Casa
Fuente: Autoría Propia

ALARMA CONTRA INCENDIOS

Exactitud	0,7
Precisión	0,74193548
Sensibilidad	0,76666667
F-Score	0,75409836

Tabla 15: Índices - Alarma contra Incendios
Fuente: Autoría Propia

TIMBRE DE ESCUELA

Exactitud	0,7
Precisión	0,80952381
Sensibilidad	0,73913043
F-Score	0,77272727

Tabla 16: Índices - Timbre de Escuela
Fuente: Autoría Propia

3.4 Resultados de la validación del modelo

Para considerar que el modelo se encuentra en condiciones, se tendrá en cuenta que la precisión, exactitud y F-Score tengan una consideración mayor o igual a 0.7 (70%).

Visto que todos los modelos presentan un F-Score mayor a 0.77, y que su precisión y exactitud están sobre la media, se puede considerar el modelo como apto para su aplicación en el mundo real, ya que la mayoría de las veces la predicción resultante que ofrecerá será correcta, y se considera un porcentaje suficiente para la construcción de un prototipo.

3.5 Conclusiones

En base a los resultados obtenidos del sistema de reconocimiento de sonidos cotidianos, se concluye que:

- Los sonidos pueden ser otro tipo de entrada para las redes predictivas, su muestreo para la aplicación en el entrenamiento de una red neuronal puede ser complejo debido a la calidad de toma de muestras en un entorno que no está dedicado a la grabación de sonido o con el equipo dedicado para el trabajo, pero con la construcción del modelo se demuestra que es posible.
- Se puede realizar un modelo de predicción de una neuronal con datos de audio obtenidos por internet desde diferentes repositorios con autoría libre para la creación de aplicaciones de soporte para personas con necesidades especiales.
- Los análisis de redes neuronales y modelos de predicción como la matriz de confusión para la extracción de índices se pueden aplicar en el proceso de aceptación de un modelo de predicción creado a partir de Teachable Machine.
- Se pudo comprobar y verificar que se puede crear una aplicación móvil haciendo uso de un modelo de Tensorflow Lite usando el método de entrenamiento de Teachable Machine, ofreciendo una aplicación con un índice de precisión y exactitud aceptables y usables en el mundo real.

BIBLIOGRAFÍA

[1] P. Shrestha, A. Sathanur, S. Maharjan, E. Saldanha, D. Arendt y S. Volkova, «Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of GitHub, Twitter and Reddit.,» *Plos One,* vol. 15, nº 3, p. 0230250, 2020.

[2] R. Crystal-Ornelas, C. Varadharajan, B. Bond-Lamberty, K. Boye, M. Burrus, S. Cholia, M. Crow, J. Damerow, R. Devarakonda, A. Goldman, S. Heinz, V. Hendrix, Z. Kakalia, S. C. Pennington, E. Robles, A. Rogers, M. Simmonds, T. Velliquette, H. Weierbach, P. Weisenhorn, J. N. Welch y D. A. Agarwal, «A Guide to Using GitHub for Developing and Versioning Data Standards and Reporting Formats,» *Earth and Space Science,* vol. 8, nº 8, 19 Julio 2021.

[3] M. Madeja, J. Porubän, S. Chodarev, M. Sulír y F. Gurbáľ, «Empirical Study of Test Case and Test Framework Presence in Public Projects on GitHub,» *Applied Sciences,* vol. 7250, nº 11, p. 7250, Agosto 2021.

[4] H.-A. Ordoñez-Erazo,, C. Ordóñez y V. Buchelli, «Recuperación y clasificación de arquitecturas software en GitHub para reutilización, soportado por ontologías,» *Revista Científica,* vol. 41, nº 2, Mayo 2021.

[5] J. Kim, J. Wi y Y. Kim, «Sequential Recommendations on GitHub Repository,» *Applied Sciences,* vol. 11, nº 1585, p. 1585, Febrero 2021.

[6] Z. Liao, X. Qi, Y. Zhang, X. Fan y Y. Zhou, «How to Evaluate the Productivity of Software Ecosystem: A Case Study in GitHub,» *Scientific Programming,* vol. 2020, p. 13, Agosto 2020.

[7] M. Carney, W. Barron, I. Alvarado y K. Phillips, «Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification,» *CHI: Conference on Human Factors in Computing Systems,* p. 1, Abril 2020.

[8] J. Hyunja, «Feasibility Study of Google's Teachable Machine in Diagnosis of Tooth-Marked Tongue,» *Journal of dental hygiene science,* vol. 20, nº 4, pp. 206-212, 2020.

[9] K. L. Kumar y A. Pja, «Automatic diagnosis of COVID-19 disease using deep convolutional neural network with multi-feature channel from respiratory sound data: Cough, voice, and breath,» *Alexandria Engineering Journal,* vol. 61, nº 2, pp. 1319-1334.

[10] N. U. Islam y S. Lee, «Interpretation of Deep CNN Based on Learning Feature Reconstruction With Feedback Weights,» *IEEE ACCESS,* vol. 7, pp. 25195-25208, 2019.

[11] H. Jiang, D. Wu, R. Jiao y Z. Wang, «Analytical Comparison of Two Emotion Classification Models Based on Convolutional Neural Networks,» *Complexity,* vol. 2021, Febrero 2021.

[12] M. Ayache, H. Kanaan, K. Kassir y Y. Kassir, «Speech Command Recognition Using Deep Learning,» *International Conference on Advances in Biomedical Engineering,* vol. 6, pp. 24-29, 2021.

[13] Y.-Y. Lin, W.-Z. Zheng, W.-Z. Chu, J.-Y. Han, Y.-H. Hung, G.-M. Ho, C.-Y. Chang y Y.-H. Lai, «A Speech Command Control-Based Recognition System for Dysarthric Patients Based on Deep Learning Technology,» *Applied Sciences,* vol. 11, nº 6, Marzo 2021.

[14] N. Chieng, M. Tariqul y M. Shahidul, «A CNN-Based Smart Waste Management System Using TensorFlow Lite and LoRa-GPS Shield in Internet of Things Environment,» *IEEE ACCESS,* vol. 9, pp. 153560-153574, 2021.

[15] S. Akilesh, R. Manthan, S. Shayantan y B. Sumedha, «Android-based object recognition application for visually impaired,» de *ITM WEB OF CONFERENCES*, 2021.

[16] T. Zebin, P. S. J., N. Peek, A. Casson y K. Ozanyan, «Design and Implementation of a Convolutional Neural Network on an Edge Computing

Smartphone for Human Activity Recognition,» *IEEE Access,* vol. 7, pp. 133509-133520, 2019.

[17] H. A. Bautista Mier, A. F. Rodriguez Gutierrez, C. Torres Espinoza y J. H. Lopez Ramirez, «Uso y percepción del personal de salud sobre una aplicación móvil para la valoración geriátrica integral,» *MEDUNAB,* vol. 24, nº 2, pp. 169-175, 2021.

[18] J. Charles Ceccato, J. C. Ceccato, M. J. Duran, D. W. Swanepoel y C. Smits, «French Version of the Antiphasic Digits-in-Noise Test for Smartphone Hearing Screening,» *Frontiers in public Health,* vol. 9, Octubre 2021.

[19] F. Zangenehnejad y Y. Gao, «GNSS smartphones positioning: advances, challenges, opportunities, and future perspectives,» vol. 2, nº 1, pp. 1-23, 2021.

[20] G. B. Satrya, «Digital Forensics Study of a Cloud Storage Client: A Dropbox Artifact Analysis,» *Commit Journal,* vol. 13, nº 2, pp. 57-66, 2019.

[21] G. Singh, M. Rohit Kumar , C. Kumar y K. Naik, «MultiDroid: An Energy Optimization Technique for Multi-Window Operations on OLED Smartphones,» *IEEE Access,* vol. 6, p. 31983, 2018.

[22] N. Abd.Rahman, J. B. Mohd Ali y M. A. Bin Baharom, «Preliminary Development of MyAras: Level 1 Mobile Virtual Lab,» *International Journal of Emerging Technologies in Learning,* vol. 14, nº 24, pp. 191-139, 2019.

[23] S. Alkhuraiji, «Design and Implementation of an Android Smart Parking Mobile Application,» *TEM JOURNAL,* vol. 9, nº 4, pp. 1357-1363, 2020.

[24] S. Yerima, M. Alzaylaee y S. Sezer, «Machine learning-based dynamic analysis of Android apps with improved code coverage,» *EURASIP JOURNAL ON INFORMATION SECURITY,* vol. 2019, nº 1, pp. 1-24, 2019.

[25] D. Gałan, K. Fisz y P. Kopniak, «A multi-criteria comparison of mobile applications built with the use of Android and Flutter Software Development Kits,» *Journal of Computer Sciences Institute,* vol. 19, nº 1, 2021.

[26] M. Cournet, E. Sarrazin, L. Dumas, J. Michel, J. Guinet, D. Youssefi, V. Defonte y Q. Fardet, «GROUND TRUTH GENERATION AND DISPARITY ESTIMATION FOR OPTICAL SATELLITE IMAGERY,» *THE INTERNATIONAL ARCHIVES OF THE PHOTOGRAMMETRY, REMOTE SENSING AND SPATIAL INFORMATION SCIENCES,* vol. 43, nº B2, pp. 127-134, 2020.

[27] K. Elmer y M. Kalacska, «A High-Accuracy GNSS Dataset of Ground Truth Points Collected within Îles-de-Boucherville National Park, Quebec, Canada,» *DATA,* vol. 6, nº 32, p. 32, 2021.

[28] B. Husham , N. Sulaiman, S. Rahman , R. Atan, S. Lailatul y M. Alghrairi, «Identification of Distributed Denial of Services Anomalies by Using Combination of Entropy and Sequential Probabilities Ratio Test Methods,» *SENSORS,* vol. 21, nº 6453, p. 6453, 2021.

[29] V. Starovoitov y Y. Golub, «About the confusion-matrix-based assessment of the results of imbalanced data classification,» *INFORMATIKA,* vol. 18, nº 1, pp. 61-71, 2021.

[30] W.-L. Wu, M.-H. Lee, H.-T. Hsu, W.-H. Ho y J.-M. Liang, «Development of an Automatic Functional Movement Screening System with Inertial Measurement Unit Sensors,» *APPLIED SCIENCES,* vol. 11, nº 96, p. 96, 2021.