



INGENIERÍA DE SONIDO

**RECONOCIMIENTO DE ESCENAS ACÚSTICAS
UTILIZANDO REDES NEURONALES E
IMPLEMENTACIÓN EN UN DISPOSITIVO MÓVIL†**

Autor: Luciano De Bortoli

Tutor: Laurence Bender

Cotutor: Leonardo Pepino

(†) Tesis para optar al título de Ingeniero de Sonido

Diciembre 2019

RESUMEN

En esta investigación se aborda la problemática de la clasificación de escenas acústicas mediante el uso de técnicas de aprendizaje automático y procesamiento de señales de audio. Se desarrolla un modelo capaz de predecir automáticamente el entorno en el que se registra una señal de audio.

Para llevar esto a cabo, se realizan pruebas con diversas arquitecturas de redes neuronales artificiales entrenadas utilizando descriptores espectrales, coeficientes cepstrales y espectrogramas. Los descriptores se obtienen a partir de una selección de señales ambientales organizadas en once clases, pertenecientes a las bases de datos TUT-2017 y TUT-2018, las cuales previamente se analizan y depuran de forma exhaustiva.

Los diferentes modelos predictivos se comparan utilizando métricas clásicas de evaluación de clasificadores. En base a los resultados obtenidos se propone un modelo final, que consiste de una red convolucional entrenada utilizando técnicas de aumento de datos sobre espectrogramas en escala Mel.

El modelo final se implementa en una aplicación para dispositivos móviles, con sistema operativo Android, la cual realiza una predicción de escena sonora en función del sonido ambiental capturado con el micrófono. Por último, se evalúa el rendimiento del modelo implementado a través de una base de datos de sonidos ambientales registrados con un único dispositivo móvil en la Ciudad de Buenos Aires, especialmente confeccionada para esta investigación, con la cual se obtiene un F-valor global superior al 55%, aunque con gran dispersión entre las once clases.

Palabras Clave: Escenas Acústicas • Redes Neuronales Artificiales • Dispositivos Móviles

ABSTRACT

This research addresses the topic of acoustic scene classification, through the use of machine learning and audio signal processing techniques. A model capable of automatically predicting the audio signal environment is developed.

To accomplish this, experiments are performed considering various artificial neural network architectures, trained using spectral descriptors, cepstral coefficients and spectrograms. The descriptors are extracted from a selection of environmental signals divided into eleven classes belonging to the TUT-2017 and TUT-2018 databases, which are previously analyzed and filtered.

These predictive models are compared by means of standard classifier evaluation metrics. Based on these results a final model is proposed, which consists of a convolutional neural network trained using data augmentation techniques on Mel scale spectrograms.

The final model is implemented in an Android application for mobile devices, which infers the acoustic environment based on the ambient sound acquired through the microphone. The implemented model is finally evaluated using an environmental sound dataset recorded using a single mobile device in the City of Buenos Aires, specifically assembled for this research, obtaining over 55% global F-score, although with substantial dispersion between the eleven classes.

Keywords: Acoustic Scenes • Artificial Neural Networks • Mobile Devices

AGRADECIMIENTOS

Esta tesis es el fruto de un trabajo de más de un año de dedicación y aprendizaje. Expreso mis agradecimientos a todos los que me acompañaron en esta etapa. En primer lugar, a las autoridades de la Universidad Nacional de Tres de Febrero, a su rector Lic. Anibal Jozami, al personal docente, no docente y a todos quienes hacen posible la educación pública, gratuita y de calidad en la Argentina. Un especial agradecimiento a mis tutores Laurence Bender y Leonardo Pepino por orientarme y brindar su apoyo, sin el cual esta tesis no habría sido posible. Por último y principal a mis padres por ser el soporte incondicional para la finalización de mis estudios y desarrollo personal.

Luciano De Bortoli

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN	1
1.1 FORMULACIÓN DEL PROBLEMA	1
1.2 OBJETIVOS	3
1.2.1 Objetivo general	3
1.2.2 Objetivos específicos	3
1.3 ESTRUCTURA Y METODOLOGÍA DE LA INVESTIGACIÓN	4
2. ESTADO DEL ARTE	5
2.1 REVISIÓN HISTÓRICA	5
2.2 PERSPECTIVAS ACTUALES	9
3. MARCO TEÓRICO	10
3.1 ESCENAS ACÚSTICAS	10
3.1.1 Definición	10
3.1.2 Bases de datos	11
3.2 REPRESENTACIÓN DE SEÑALES DIGITALES	14
3.2.1 Representación del sonido	14
3.2.2 Descriptores de audio	16
3.3 APRENDIZAJE AUTOMÁTICO	19
3.3.1 Introducción	19
3.3.2 Preprocesamiento de datos	20
3.3.3 Modelos	21
3.3.4 Redes neuronales artificiales profundas	23
3.3.5 Redes neuronales convolucionales	25
3.3.6 Entrenamiento de redes neuronales	27
3.3.7 Evaluación de clasificadores	29
4. METODOLOGÍA	31
4.1 BASES DE DATOS	31
4.1.1 Selección y procesamiento	31
4.1.2 Combinaciones	33
4.1.3 Base de datos de evaluación	34
4.2 DESCRIPTORES	35
4.2.1 Método de extracción	35
4.2.2 Selección de atributos	36
4.3 MODELO PREDICTIVO	37
4.3.1 Selección de arquitectura de red	37
4.3.2 Modelo final propuesto	39
4.3.3 Técnicas de aumento de datos	40

4.4 APLICACIÓN EN ANDROID	42
4.4.1 Adquisición de audio	42
4.4.2 Procesamiento de señales	43
4.4.3 Inferencia	45
4.4.4 Estructura de interfaz de usuario	46
5. RESULTADOS	47
5.1 PRUEBAS CON DESCRIPTORES	47
5.2 PRUEBAS CON MODELOS	51
5.2.1 Pruebas con redes convolucionales	51
5.2.2 Pruebas con aumento de datos	53
5.3 EVALUACIÓN FINAL	54
6. DISCUSIÓN	56
6.1 COMENTARIOS SOBRE LAS BASES DE DATOS	56
6.2 DESCRIPTORES	57
6.2.1 Análisis de valores medios	57
6.2.2 Análisis de métricas	58
6.3 MODELOS	58
6.3.1 Análisis de arquitecturas CNN	58
6.3.2 Análisis de filtros convolucionales	59
6.3.3 Análisis de inter-validación de datos	59
6.3.2 Análisis de pruebas con aumento de datos	59
6.3.4 Análisis de pruebas finales	60
6.4 COMENTARIOS SOBRE LA EVALUACIÓN DE LA APLICACIÓN	61
7. CONCLUSIONES	62
8. TRABAJOS FUTUROS	63
BIBLIOGRAFÍA	64
ANEXOS	72
ANEXO I: BASES DE DATOS Y DESCRIPTORES DE ESCENAS SONORAS	72
ANEXO II: RESULTADOS COMPLETOS DE PRUEBAS	75
ANEXO III: MODELO FINAL IMPLEMENTADO EN ANDROID	85
ANEXO IV: CÓDIGO IMPLEMENTADO	87

ÍNDICE DE FIGURAS

Nº	Descripción	pág
Figura 1	Diferenciación entre escenas acústicas y eventos acústicos	10
Figura 2	Proceso de enventanado de una señal para el cálculo de STFT	15
Figura 3	Escala Mel Slaney de 128 componentes con rango de frecuencia 0-20 KHz	17
Figura 4	Banco de filtros triangulares Mel Slaney para 128 componentes	18
Figura 5	Diagrama de flujo de aprendizaje supervisado para un modelo clasificador	21
Figura 6	(a) Esquema de neurona. (b) Ejemplo de estructura de capas	23
Figura 7	Funciones de activación comúnmente utilizadas para redes neuronales artificiales ...	24
Figura 8	Ejemplos de las operaciones de convolución 2D y agrupamiento de máximos	26
Figura 9	Esquema ilustrativo de una CNN para clasificación	26
Figura 10	Matriz de confusión para modelo binario (izq) y multiclas (der)	30
Figura 11	Ejemplos de análisis ROC para diversas distribuciones binarias	30
Figura 12	Espectros promediados tras la aplicación de procesos de bases de datos	32
Figura 13	Combinación de bases de datos utilizadas para los experimentos con modelos ANN .	33
Figura 14	Influencia de N y H en la resolución frecuencial (a) y temporal (b) del espectrograma	35
Figura 15	Esquema para la combinación de descriptores de escenas acústicas	36
Figura 16	Variaciones de DNN para la selección de descriptores de escenas acústicas	36
Figura 17	Esquema de diversas arquitecturas de CNN empleadas	38
Figura 18	Esquema completo de modelo propuesto para la clasificación de escenas acústicas .	39
Figura 19	Ejemplos de aumento de datos utilizando enmascaramiento sobre espectrogramas .	40
Figura 20	Generación aleatoria de filtros Butterworth en escala Mel normalizada	41
Figura 21	Ejemplo de aumento de datos por filtrado aleatorio de espectrogramas	41
Figura 22	Esquema de procesamiento y cómputo de espectrograma en Android	43
Figura 23	Validación de metodología para extracción de espectrogramas en Android	44
Figura 24	Esquema de implementación de red neuronal en Android	45
Figura 25	Esquema de interacción entre objetos de la interfaz de usuario y procesos de fondo	46
Figura 26	Medias de espectro Mel, MFCC y SCO por frecuencia para TUT-17 (a) y TUT-18 (b)....	47
Figura 27	Medias y desvíos estándar para descriptores globales de las clases de las bases TUT .	48
Figura 28	Comparación de F-valor por clase para DNN-0 utilizando descriptores de bases TUT ..	49
Figura 29	F-valor por clase para diversas arquitecturas DNN utilizando MFCC y espectro Mel	50
Figura 30	F-valor por clase para diversas arquitecturas de redes convolucionales	51
Figura 31	F-valor por clase para variantes de modelos CNN con filtros verticales y horizontales	52
Figura 32	F-valor por clase para arquitectura CNN-C utilizando técnicas de aumento de datos ..	53
Figura 33	F-valor por clase para evaluación de variantes del modelo final utilizando BA-DB	54
Figura 34	Matrices de confusión para evaluación de variantes de aumento de datos	55
Figura 35	Curva ROC y AUC para modelo final utilizando técnica de filtros aleatorios	55

Nº	Descripción	pág
Figura I.1	Balance de bases de datos más importantes para clasificación de escenas sonoras	72
Figura I.2	Comparación de medias de envolvente espectral para bases de datos estudiadas	72
Figura II.1	Matrices de confusión para pruebas preliminares 1 sobre la base de datos TUT-2017 .77	
Figura II.2	Matrices de confusión para pruebas preliminares 1 sobre la base de datos TUT-2018 .78	
Figura II.3	Ánalysis de predicciones del modelo final sobre escena de automóvil de BA-DB82	
Figura II.4	Ánalysis de predicciones del modelo final sobre escena de autobús de BA-DB82	
Figura II.5	Ánalysis de predicciones del modelo final sobre escena de café de BA-DB83	
Figura II.6	Ánalysis de predicciones del modelo final sobre escena de patio de comidas de BA-DB 83	
Figura II.7	Ánalysis de predicciones del modelo final sobre escena de habitación de BA-DB84	
Figura II.8	Ánalysis de predicciones del modelo final sobre calle urbana de BA-DB84	
Figura III.1	Progreso de entrenamiento automático de modelo final implementado86	
Figura III.2	Interfaz de usuario de aplicación desarrollada en Android86	
Figura IV.1	PYTHON: funciones definidas para extracción de descriptores de escenas sonoras87	
Figura IV.2	PYTHON: combinación de base de datos para el entrenamiento del modelo87	
Figura IV.3	PYTHON: arquitectura de red neuronal para clasificador de escenas sonoras88	
Figura IV.4	PYTHON: procesos de aumento de datos definidos para lote de espectrogramas89	
Figura IV.5	JAVA: Inicialización de variables globales para la aplicación Android desarrollada90	
Figura IV.6	JAVA: extracto para la adquisición de muestras de audio en aplicación de Android90	
Figura IV.7	JAVA: hilo de procesamiento dedicado a la extracción de espectrogramas91	
Figura IV.8	JAVA: hilo de procesamiento dedicado a la inferencia de red neuronal91	
Figura IV.9	JAVA: inicialización de clase para la extracción de espectrogramas en Android.92	
Figura IV.10	JAVA: funciones para el cómputo de transformada rápida de Fourier en Android92	
Figura IV.11	JAVA: funciones para el cómputo del espectrograma en escala Mel en Android93	
Figura IV.12	JAVA: funciones traducidas de Numpy para operaciones de matrices en Android94	

ÍNDICE DE TABLAS

Nº	Descripción	pág
Tabla 1	Revisión del estado del arte sobre clasificación de escenas acústicas	8
Tabla 2	Recopilación de Bases de datos para clasificación de escenas y eventos acústicos	12
Tabla 3	Clases de las principales bases de datos para clasificación de escenas acústicas	13
Tabla 4	Ecuaciones de descriptores según aparecen en la literatura	16
Tabla 5	Locaciones de Buenos Aires seleccionadas para registrar las señales de evaluación	34
Tabla 6	Configuración de parámetros de AudioRecord para adquisición de audio en Android	42
Tabla 7	F-valor global y mínimo para descriptores de bases TUT utilizando DNN-0	49
Tabla 8	F-valor global para arquitecturas DNN utilizando MFCC y espectro Mel	50
Tabla 9	F- valor global y mínimo para diversas arquitecturas CNN utilizando base de datos TUT	51
Tabla 10	F-valor global de modelos utilizando filtros convolucionales verticales y horizontales ...	52
Tabla 11	Resultados de aciertos (%) utilizando CNN-C para inter-validación de bases de datos ...	52
Tabla 12	F-valor y AUC para modelo CNN-C utilizando técnicas de aumento de datos	53
Tabla 13	Métricas globales de evaluación para variantes de modelo final utilizando BA-DB	54
Tabla I.1	Clasificación de descriptores utilizados para voz (V), música (M) y ambiente (A)	73
Tabla II.1	F-valor por clase para pruebas con descriptores utilizando DNN-0	75
Tabla II.2	F-valor por clase para las pruebas con MFCC y Espectro Mel	76
Tabla II.3	F-valor por clase para pruebas con arquitecturas de CNN	79
Tabla II.4	F-valor por clase para pruebas con aumento de datos	80
Tabla II.5	F-valor por clase según tipo de filtros convolucionales	81
Tabla II.6	F-valor por clase con diversas técnicas de entrenamiento utilizando BA-DB	81
Tabla II.7	Resultados de métricas por clase para el modelo final utilizando BA-DB	81
Tabla III.1	Detalle de arquitectura de red neuronal convolucional implementada	85

LISTA DE ACRÓNIMOS

INGLÉS	SIGLA	ESPAÑOL
Autocorrelation Function	ACF	Función de Autocorrelación
Automatic Music Transcription	AMT	Transcripción Musical Automática
Artificial Neural Networks	ANN	Redes Neuronales Artificiales
Auditory Saliency Map	ASM	Mapa de Prominencia Auditiva
Automatic Speech Recognition	ASR	Reconocimiento de Voz Automático
Audio Waveform	AW	Forma de Onda
Code Excited Linear Prediction	CELP	Predicción lineal con excitación por código
Convolutional Neural Network	CNN	Red Neuronal Convolucional
Deep Neural Network	DNN	Redes Neuronales Profundas
Event-Modulated Noises	EMN	Ruidos Modulados por Eventos
Gaussian Mixture Models	GMM	Modelo de Mezclas Gaussianas
Head Related Transfer Function	HRTF	Función de Transferencia de la Cabeza
Instrument Recognition	IR	Reconocimiento de Instrumentos
K-Nearest Neighbours	kNN	K-Vecinos Más Próximos
Latent Acoustic Topic	LATEA	Tópico Acústico Latente
Local Binary Pattern	LBP	Patrón Binario Local
Latent Dirichlet Allocation	LDA	Asignación de Dirichlet Latente
Linear Predictive Cepstral Coefficients	LPCC	Coeficientes cepstrales de predicción lineal
Long short-term memory	LSTM	Memoria de Corto y Largo Plazo
Mel Frequency Cepstral Coefficients	MFCC	Coeficientes cepstrales de frecuencia Mel
Music Information Retrieval	MIR	Recuperación de Información Musical
Machine Learning	ML	Aprendizaje Automático
Matching Pursuit	MP	Búsqueda Coincidente
Narrow Band Autocorrelation Function	NB-ACF	Autocorrelación de Banda Estrecha
Non-negative matrix factorization	NMF	Factorización No Negativa de Matrices
Principal Components Analysis	PCA	Análisis de Componentes Principales
Recurrent Convolutional Neural Networks	RCNN	Red Neuronal Convolucional Recurrente
Recurrent Deep Neural Network	RDNN	Red Neuronal Profunda Recurrente
Recurrent Neural Network	RNN	Red Neuronal Recurrente
Spectral Centroid	SC	Centroide Espectral
Spectral Flatness	SF	Planicidad Espectral
Speaker Recognition	SR	Reconocimiento de Orador
Spectral Roll-off Point	SRO	Rodamiento Espectral
Subband Spectral Flux	SSF	Flujo Espectral de Subbanda
Support Vector Machines	SVM	Máquinas de Soporte Vectorial
T-Distributed stochastic neighbor embedding	TSNE	Incrustación Estocástica Distribuida T
Zero Crossing Rate	ZCR	Tasa de Cruce por Cero

1. INTRODUCCIÓN

1.1 FORMULACIÓN DEL PROBLEMA

En este trabajo se aborda el desarrollo de un sistema diseñado para el reconocimiento de escenas acústicas cotidianas y su implementación en un dispositivo móvil. Con este fin, se realiza un estudio de las técnicas de análisis de escenas acústicas utilizadas para identificar un entorno sonoro mediante diversos descriptores, permitiendo diferenciarlo de otros. Entre estas técnicas, se destaca el aprendizaje automático (ML), que consiste en el diseño de algoritmos y modelos estadísticos para ejecutar tareas específicas a partir del reconocimiento de patrones sobre una base de datos de entrenamiento.

En trabajos previos [1-5], se han implementado técnicas de aprendizaje automático enfocadas en el reconocimiento de voz o la clasificación musical, dado que las señales de voz y música suelen presentar patrones frecuenciales y temporales reconocibles por su estructura. Si bien las escenas acústicas y los sonidos ambientales también presentan patrones reconocibles, éstos constituyen un mayor desafío debido a que carecen de estructura definida como son las reglas gramaticales, semánticas y lógicas de un idioma, o bien la ritmica y armonía de una pieza musical. Sin embargo, en los últimos años han surgido nuevos descriptores específicamente diseñados para la clasificación de escenas acústicas, aunque el diseño de modelos robustos sigue siendo un problema abierto.

Con el desarrollo de nuevas tecnologías, los sistemas inteligentes muestran una tendencia a automatizar procesos internos en función de las necesidades del usuario, sin requerir una configuración manual por parte del mismo. La auto-configuración de parámetros a partir de la detección del entorno sonoro puede ser muy útil en múltiples aplicaciones, aunque en la actualidad no es muy habitual.

Una posible aplicación sería en el caso de las tabletas o teléfonos móviles, los cuales podrían determinar el entorno en el que se encuentra el usuario (*context awareness*) a partir de técnicas de clasificación de escenas acústicas, pudiendo utilizarse para alterar el nivel de timbre en espacios ruidosos, activar el modo vibrador en una sala de conferencias automáticamente, o bien en proyectos de ciudades inteligentes midiendo el nivel de polución sonora, entre otros [6].

Por otra parte, con el crecimiento del desarrollo de vehículos autónomos inteligentes, se vislumbran campos de aplicación para el sector automotriz en los que el reconocimiento de escenas acústicas podría complementar los sistemas de visión actualmente implementados. De forma similar, en el campo de la robótica, el reconocimiento de escenas acústicas podría otorgarles a los robots capacidades básicas de percepción auditiva para interactuar con su entorno, en conjunto con el uso de tecnologías de localización de sonidos a partir del aprendizaje supervisado de la Función de Transferencia Relacionada con la Cabeza (HRTF) [7].

En todas estas aplicaciones es de suma utilidad contar con sistemas que determinen el entorno del usuario para poder adaptarse de forma automática. Para llevar esto a cabo, el primer paso es desarrollar un algoritmo capaz de reconocer un grupo determinado de escenas acústicas con alto grado de exactitud. Si bien la investigación sobre clasificación automática de sonidos ha crecido considerablemente en los últimos años, su aplicación en dispositivos móviles es limitada y muy dependiente de la disponibilidad de bases de datos de escenas acústicas adecuadas para confeccionar modelos.

El objetivo de esta tesis es crear un clasificador capaz de determinar la escena sonora en donde se registra una señal en base al análisis del audio captado por el micrófono de un dispositivo móvil. Se desarrolla un modelo a partir del entrenamiento supervisado de una red neuronal artificial, en el que se hace uso de bases de datos de sonidos ambientales de acceso público, utilizando diversas técnicas de extracción de descriptores, y posteriormente se valida el sistema a través de una serie de pruebas que determinan el desempeño de la tarea de clasificación. El algoritmo debe ser evaluado en las condiciones de uso normales, es decir, utilizando sonidos registrados con dispositivos móviles representativos de las escenas acústicas.

1.2 OBJETIVOS

1.2.1 Objetivo general

Desarrollar un sistema de clasificación automática de escenas acústicas para implementar en un dispositivo móvil, el cual permita identificar los entornos sonoros más usuales: espacios urbanos, recintos públicos e interiores de medios de transporte.

1.2.2 Objetivos específicos

- Recopilar trabajos previos asociados a las escenas acústicas.
- Recopilar bases de datos de escenas acústicas.
- Recopilar técnicas de parametrización.
- Extraer descriptores de sonidos ambientales.
- Desarrollar la estructura del modelo predictivo computarizado.
- Entrenar y optimizar el modelo.
- Implementar el algoritmo en una aplicación Android.¹
- Desarrollar una base de datos de evaluación con sonidos de Buenos Aires.
- Evaluar el modelo utilizando la base de datos de evaluación.
- Presentar los resultados y conclusiones.

¹ El código de la aplicación de Android es públicamente accesible en el siguiente repositorio de GitHub:
<https://github.com/lucianodebortoli/SceneClassifier/tree/master/android/SceneClassifier>

1.3 ESTRUCTURA Y METODOLOGÍA DE LA INVESTIGACIÓN

En lo referente a la metodología de la investigación, inicialmente se realiza una revisión y recopilación exhaustiva de la literatura científica, a modo de establecer el estado del arte y construir un marco teórico de la disciplina. De aquí se extraen las herramientas utilizadas en la etapa de desarrollo. Luego, se determina el alcance de la investigación, estableciendo las escenas acústicas que se pretenden diferenciar, teniendo en cuenta los elementos necesarios para el desarrollo del algoritmo, entre ellos, la base de datos, las técnicas de parametrización y la estructura del modelo a utilizar. Finalmente, se procede a analizar el rendimiento del algoritmo propuesto y evaluar su implementación final en una plataforma móvil.

El enfoque de la investigación es cuantitativo ya que aspira estudiar los procesos que intervienen en la clasificación de escenas acústicas, donde se evalúa la capacidad de acierto del sistema que se desarrolla para el reconocimiento automático de entornos sonoros, midiendo la cantidad porcentual de clasificaciones exitosas por sobre el total de pruebas realizadas.

La presente investigación se estructura en secciones, las cuales se detallan a continuación. En la sección 1 se introduce la problemática, los objetivos generales y específicos de la tesis. En la sección 2 se desarrolla el estado del arte, detallando los trabajos previos y las perspectivas actuales. En la sección 3 se profundiza el marco teórico, definiendo los tópicos relativos al aprendizaje automático, a las escenas acústicas y a la parametrización de descriptores. En la sección 4 se explica la metodología aplicada en el desarrollo del modelo utilizado en el reconocimiento automático de escenas acústicas y su implementación en el sistema operativo Android. En la sección 5 se muestran los resultados obtenidos tras la evaluación del modelo y la aplicación móvil, utilizando métricas varias. En la sección 6 se discuten los resultados obtenidos, analizando el desempeño del algoritmo final y de su implementación en un dispositivo móvil. En la sección 7 se enuncian las conclusiones obtenidas a partir de los resultados de este trabajo. Finalmente, en la sección 8 se presentan posibles perspectivas para trabajos futuros.

2. ESTADO DEL ARTE

A través de la revisión de artículos científicos, trabajos de tesis y otras contribuciones académicas relacionadas a la disciplina, se establece el estado del arte, el cual consiste en una compilación de resultados obtenidos a partir de las investigaciones anteriores. En esta sección se hace una revisión histórica de los antecedentes relativos al reconocimiento de escenas acústicas.

2.1 REVISIÓN HISTÓRICA

Los sistemas de reconocimiento de sonidos tienen su origen a mediados del siglo XX cuando las empresas de comunicación comenzaron a realizar investigaciones en el ámbito forense, originalmente relativas al reconocimiento de la voz, lo que luego evolucionaría en lo que hoy se conoce como *reconocimiento automático de palabras* (ASR) y *reconocimiento de orador* (SR).

A finales de siglo, aparecen nuevas técnicas de parametrización y modelos acústicos predictivos computarizados, al mismo tiempo que crece la demanda de nuevas herramientas para el reconocimiento automático del sonido, como por ejemplo la música. Entre las disciplinas más destacadas, se encuentra la *recuperación de información musical* (MIR), que comprende técnicas como el *reconocimiento de notas musicales* (AMT), la *identificación de instrumentos musicales* (IR) y la identificación de géneros musicales.

La clasificación de escenas acústicas junto a la detección de eventos sonoros comienza a tomar relevancia a medida que las tecnologías digitales evolucionan. Esta disciplina hace uso de técnicas basadas en métodos previamente utilizados para voz y la música. A partir del nuevo milenio, surgen varios estudios de reconocimiento de escenas acústicas y clasificación de sonidos ambientales utilizando nuevos descriptores y modelos matemáticos.

El estado del arte referido a la disciplina ha sido relevado y sintetizado en diversas revisiones bibliográficas, entre las que se destacan las de Virtanen et al. [8], Alías et al. [9], Barchiesi et al. [10] y Stowell et al. [11].

Los primeros trabajos se basan en el uso de descriptores matemáticos que habían sido exitosos para el reconocimiento de voz, como es el caso de los *coeficientes cepstrales de frecuencia Mel* (MFCC). Beritelli et al. [12], los implementan en un sistema de clasificación de sonidos ambientales. Debido a su éxito, los MFCC se convierten entonces en la vara a superar por los investigadores de la época.

Con el fin de proponer descriptores novedosos para suplantar los MFCC, surgen nuevos trabajos, como es el caso de Muhammad et al. [13], quienes introducen el uso de una serie de 17 descriptores MPEG-7 temporales y espectrales. Estos autores sugieren que el rendimiento de esta técnica representa un avance sobre el uso de MFCC, aunque reconociendo que los mejores resultados se obtienen al combinar los nuevos descriptores con MFCC y la función de tasa de cruce por cero (ZCR). Un nuevo conjunto de descriptores es propuesto por Tsau et al. [14], basados en técnicas de *predicción lineal con excitación por código* (CELP) en el contexto de una red de clasificación bayesiana. Por su parte, Valero et al. [15] introducen otro conjunto de descriptores que consideran las características temporales, espectrales y perceptuales del análisis de las escenas acústicas, específicamente las relativas a la función de autocorrelación (ACF) aplicada en conjunto a un banco de filtros Mel realizando experimentos sobre audios de 4 horas de duración de 15 diferentes escenas acústicas. Por último, se encuentran los histogramas, utilizados por Heittola [16] y Rakotomamonjy [17], en todos los casos obteniendo resultados superadores a los MFCC.

Entre otras formas de representación de señales de audio para aplicaciones de reconocimiento, se encuentra el algoritmo de *búsqueda coincidente* (MP), propuesto por Chu et al. [18] en su trabajo con sonidos de insectos y lluvia; o bien el modelo basado en un *mapa de prominencia auditiva* (ASM) propuesto por De Coensel et al. [19] en su trabajo sobre la percepción de paisajes sonoros de ruido de transporte.

Estos primeros trabajos utilizan algoritmos y modelos sencillos como la *factorización no negativa de matrices* (NMF), implementada por Cauchi [20] en su trabajo de clasificación sonora de estaciones de ferrocarril, o también las *máquinas de soporte vectorial* (SVM), utilizadas por Geiger et al. [21] para la clasificación de 10 escenas acústicas.

Se destacan también el análisis de *componentes latentes probabilísticos invariantes por desplazamiento* (SIPPCA), utilizado por Benetos et al. [22] para sonidos de estaciones de tren, clasificados en 6 categorías, y el *patrón binario local* (LBP) originalmente desarrollado para el análisis de imágenes, utilizado por Kobayashi et al. [23] y por Battaglino et al. [24] para el reconocimiento automático de entorno en una aplicación para dispositivos móviles.

No obstante, los trabajos más recientes se basan en modelos de aprendizaje profundo, influenciados por el apogeo de las *redes neuronales profundas* (DNN), que son destacadas por Sigtia et al. [25] como las de mejor rendimiento en relación al costo computacional e implementadas para reconocimiento de sonidos ambientales por Petetin et al. [26]. Otras variantes son las *redes neuronales convolucionales* (CNN), originalmente desarrolladas para procesos sobre imágenes, que comienzan a ser implementadas para sonidos ambientales, como en los trabajos de Piczak [27] y el de Tokozume [28].

Finalmente, surgen arquitecturas de redes que incorporan capas de *memoria de corto y largo plazo* (LSTM), como las utilizadas por Bae et al. [29] aplicadas sobre espectrogramas de la base de datos de escenas acústicas TUT-2016 [30]. Las LSTM son un tipo de *redes neuronales recurrentes* (RNN), las cuales introducen el concepto de memoria al trabajar con información secuencial. Se destaca el exhaustivo trabajo de Wei Dai [31] para la clasificación de quince locaciones, donde compara las RNN con modelos GMM, SVM, DNN y CNN, mostrando que su rendimiento es muy dependiente de la selección de descriptores, valorando los modelos temporales para una reducida cantidad de descriptores y los no-temporales para cantidades mayores. Los últimos trabajos se enfocan en el uso de redes neuronales en conjunto a diversas técnicas de aumento de datos (*data-augmentation* en inglés), como en el trabajo de Salamon et al. [32].

En la Tabla 1 se sintetizan los trabajos previos relacionados con la extracción de descriptores e implementación de modelos para la clasificación automática de escenas acústicas.

Tabla 1: Revisión del estado del arte sobre clasificación de escenas acústicas

Año	Referencia	Método	Ref.
2001	Gygi.B et al.	Ruidos modulados por eventos	[33]
2001	Ando. Y	Función de autocorrelación	[34]
2002	Peltonen. V. et al.	BW, ZCR, SER, SF, SC, SRP, STE, LPCC, MFCC	[35]
2006	Cai. R et al.	Descriptores Espectrales, SSF, HMM, GN	[36]
2008	Beritelli. F et al.	MFCC y redes neuronales artificiales	[12]
2009	Chu. S et al.	Descriptores de Gabor, MFCC, LPCC, MP	[18]
2009	Muhammad. G et al.	HR, SC, SRP, SF, STE, LAT, ASE, HS, NB-ACF	[13]
2010	De Coensel et al.	Mapa de prominencia auditiva	[19]
2010	Heittola. T et al.	Histogramas	[16]
2011	Tsau et al.	Predicción lineal excitado por código	[14]
2011	Cauchi. B et al.	Factorización no negativa de matrices	[20]
2012	Benetos. E et al.	Modelos latentes de desplazamiento invariante	[22]
2012	Valero. X et al.	HR, SC, SRP, SF, STE, TC, LAT, ASE, HS, ACF	[15]
2012	Lee. K et al.	Descriptores escasos & máquina de Boltzmann	[37]
2012	Imoto. K. et al.	Tópico acústico latente	[38]
2013	Geiger. J et al.	Espectro Mel & SVM	[21]
2013	Dennis. J et al.	Distribución de potencia sub-banda & KNN	[39]
2014	Kobayashi. T et al.	Patrón binario local & Normalización Hellinger	[23]
2015	Petetin. Y et al.	Modelo de Mezclas Gaussianas & SVM	[26]
2015	Piczak. K. J et al.	Redes neuronales convolucionales	[27]
2015	Battaglino. D et al.	Patrón binario local	[40]
2015	Bisot. V et al.	Distribución de potencia sub-banda	[41]
2015	Rakotomamonjy. A et al.	Transformada de dispersión Gammatone	[42]
2016	Lostanlen. V et al.	Descriptores binaurales & SVM	[43]
2016	Bae. S et al.	Espectrogramas, CNN & LTSM	[29]
2016	Bisot. V et al.	Factorización no negativa de matrices & PCA	[44]
2016	Valenti. M et al.	Espectrogramas & CNN	[45]
2016	Xu. Y et al.	Regresión de etiquetas múltiples & DNN	[46]
2016	Dai. W et al.	MFCC, SVM & modelo de mezclas gaussianas	[31]
2016	Dai. W et al.	Espectrogramas, CNN, RCNN, RNN, RDNN	[31]
2016	Dai. W et al.	Descriptores Smile-6k & DNN	[31]
2016	Pillos. A et al.	MFCC, ZCR, SF, SC, MP.	[49]
2017	Salamon. J et al.	Datos Aumentados & CNN	[32]
2017	Tokozume. Y et al.	Redes neuronales convolucionales	[28]
2018	Pires. I.M et al.	MFCC & Redes neuronales profundas	[51]
2018	Defagot. G. A	MFCC, Clustering, PCA, LDA & TSNE	[47]

2.2 PERSPECTIVAS ACTUALES

Debido a los recientes aportes de muchos investigadores de la ciencia informática, y también en gran medida a las competencias anuales “Detection and Classification of Acoustic Scenes and Events” (DCASE) iniciadas en 2013, existe una gran cantidad de trabajos sobre técnicas de parametrización y modelos para la clasificación automática de escenas acústicas y eventos sonoros. Sin embargo, por lo general están concebidos para una aplicación determinada, como la detección de sonidos específicos en registros de audio, y usualmente no contemplan la implementación en dispositivos móviles. Los modelos sobresalientes de las competencias en los últimos años suelen ser una fusión de clasificadores, usualmente compuesto de redes neuronales convolucionales.

Por el momento, en la Argentina, las contribuciones en lo que refiere a clasificación de escenas acústicas son escasas, con excepción del trabajo de Defagot [47] presentado en 2018 en la Universidad de Buenos Aires, en la que usa un modelo de *asignación Dirichlet latente* (LDA) con técnicas de reducción de dimensionalidad como PCA e *incrustación estocástica de vecinos con distribución “t”* (T-SNE), para la construcción de un diccionario acústico en base a MFCCs extraídos de bases de datos de la BBC y TUT-2017.

Entre los antecedentes de implementación en aplicaciones móviles, se encuentran los trabajos de Mattia et al. [48] y el de Pillos et al. [49]. Este último, enfocado al reconocimiento automático de eventos acústicos, donde utilizan la base de datos ESC-10 [50], extrayendo descriptores tales como MFCC, ZCR, planicidad espectral (SF) y centroide espectral (SC), que luego ingresan a un perceptrón multicapa, implementado una versión reducida de la plataforma *Weka*.

En otro trabajo relacionado al campo de Reconocimiento de Actividad Humana (HAR), Pires et al. [51] se benefician de los sensores presentes en dispositivos móviles, tales como acelerómetros, magnetómetros, giroscopios y GPS. Utilizan una DNN sobre 26 MFCCs en el caso de la señal proveniente del micrófono obteniendo buenos resultados tanto en la detección de actividades (dormir, caminar, correr) como en la clasificación de escenas acústicas (cocina, calle y biblioteca). Sin embargo, no explicitan si el sistema desarrollado es evaluado en casos de uso real.

3. MARCO TEÓRICO

3.1 ESCENAS ACÚSTICAS

3.1.1 Definición

Cuando se habla de “escena”, independientemente del contexto, generalmente se refiere a un lugar en donde ocurre una acción continua, ya sea real o ficcional. Naturalmente se la concibe como el entorno en donde transcurren los hechos. Una escena acústica no es la excepción, aunque en este caso, el entorno se describe desde una perspectiva exclusivamente sonora.

Las escenas acústicas refieren entonces a sonidos representativos de una locación particular. Son sonidos familiares que recrean espacios físicos de la cotidianidad, subjetivamente reconocibles y diferenciables. Algunos ejemplos de escenas acústicas serían: un restaurante, una estación de tren, una oficina, el interior de un autobús, etc. Estos sonidos son complejos en la medida que se componen de varios elementos sonoros individuales provenientes de todas las direcciones, que se superponen en un punto de captura determinado en el espacio, y que varían en el tiempo de manera impredecible.

Es importante notar que las escenas acústicas se diferencian de lo que se conocen como “eventos acústicos”. Estos últimos, refieren a sonidos más específicos y de corta duración, generalmente asociados a una fuente sonora definida, como puede ser una explosión, un disparo, un aplauso, etc. A diferencia de los eventos acústicos, dos escenas acústicas no pueden tener lugar en simultáneo; de forma que a un segmento de audio le corresponde una sola etiqueta, como se ilustra en la Figura 1.

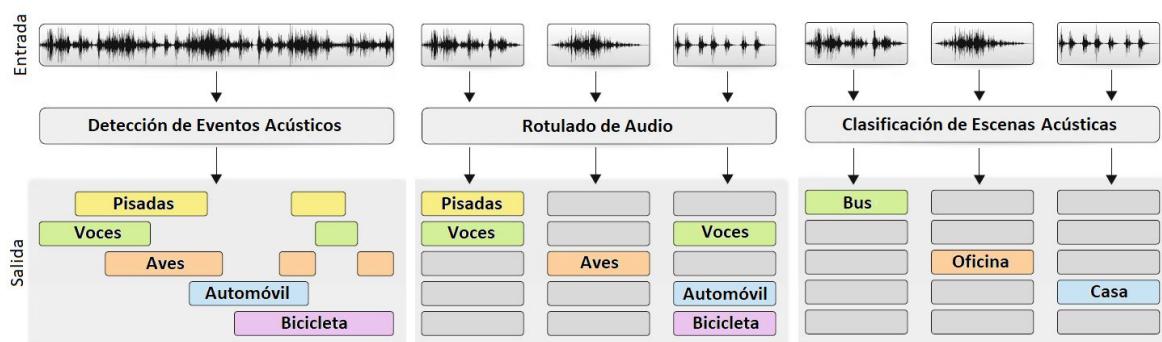


Figura 1: Diferenciación entre escenas y eventos acústicos. Adaptado de [8].

3.1.2 Bases de datos

En términos generales, el procedimiento para construir una base de datos de escenas acústicas consiste en registrar mediante un micrófono portátil diferentes entornos durante un tiempo prolongado. A diferencia de la voz, en donde usualmente se buscan audios libres de silencios y ruidos, en el contexto de escenas acústicas la mayoría contiene pasajes con todo tipo de silencios y ruidos, ya que los mismos son considerados inherentes a la escena sonora. No obstante, el investigador es capaz de intervenir la base de datos en función de la aplicación.

Como sugiere Ellis [8], una base de datos adecuada para los métodos modernos de clasificación automática de escenas acústicas debe contar con:

- Clases de escenas acústicas cotidianas, obtenidas en locaciones representativas.
- Gran cantidad de grabaciones para cada una de las escenas (tamaño).
- Variedad entre audios de la misma clase (diversidad).
- Buena distribución de audios por clase (balance).
- Audios correctamente etiquetados (desambiguación).

La recopilación de información para construir una base de datos de escenas acústicas y sus respectivas anotaciones suele ser una tarea tediosa que limita la investigación sobre el audio ambiental. Hoy en día, existe un número limitado de bases de datos de gran tamaño, disponibles de forma abierta y de fácil acceso. Gracias al trabajo de recopilación de información para clasificación de audio realizado por Heittola y otros [52-54], es posible contar con el acceso a bases de datos abiertas o comerciales para la investigación sobre clasificación de escenas y eventos acústicos, las cuales se listan en la Tabla 2.

Las bases de datos disponibles dedicadas a escenas acústicas contienen un promedio de 15 categorías o clases diferenciadas, mientras que en las dedicadas exclusivamente a eventos acústicos se pueden encontrar más de 80, de las cuales muchas contienen instancias de clases que podrían ser utilizadas para escenas acústicas. Por ejemplo, *AudioSet* de Google se destaca por su tamaño, aunque presenta un gran desbalance de audios por clase y sus etiquetas se obtienen de forma automatizada, resultando poco confiables.

Tabla 2: Recopilación de Bases de datos para clasificación de escenas acústicas.

Año	Proveedor	Nombre de Base de Datos	Clases	Audios	Duración [min]
2006	TUT	CASR	27	225	-
2006	UEA	Noise DB / Series 2	12	35	40
2006	UEA	Noise DB / Series 1	10	10	175
2009	TUT	CASA 2009	10	103	1133
2009	Dares	G1	28	123	123
2010	TUT	CASA 2010	13	160	533
2013	IEEE AASP	Scene 2013	10	100	62
2015	AucoDefr07	AucoDefr07	4	16	252
2015	LITIS	Rouen audio scene	19	3026	1513
2016	TUT	Acoustic scenes 2016	15	1170	585
2017	TUT	Acoustic scenes 2017	15	4680	780
2017	UCSD	ExtraSensory Dataset	51	302177	-
2018	TUT	Acoustic Scenes 2018 Mobile	10	10080	1680
2018	TUT	Acoustic Scenes 2018	10	8640	1440
2019	TUT/TAU	Urban Acoustic Scenes 2019	10	14400	2400
2019	Whisper	Hipster Ambient Mixtures Noise	-	2800	4800

Entre las bases de datos más utilizadas en los últimos trabajos se encuentra TUT-2017 [55], desarrollada para las competencias DCASE de 2017. La base contiene 15 clases diferentes, grabadas con micrófonos Soundman OKM II Klassik, Roland Edirol R-09 y un par de electret binaurales, utilizando una frecuencia de muestreo de 44.1 KHz a razón de 24 bits por muestra. Los autores remarcan que, si bien existen distorsiones en ciertas grabaciones, esto no debería tener un gran impacto en los resultados, basándose en competencias anteriores.

En las competencias DCASE del año siguiente, se publica una nueva base de datos llamada TUT-2018 [56], cuyos audios se registran durante febrero del 2018 en las ciudades de Barcelona, Helsinki, Londres, París, Estocolmo y Viena. En este caso se utilizan los micrófonos Soundman y Zoom F8 y electret binaurales, grabando a 48 KHz - 24 bit; y por otro lado, incorporan el uso de los celulares tales como Samsung Galaxy S7 y Iphone SE, grabando a 44.1KHz - 16 bits. De esta manera, la competencia hace pública dos bases de datos en simultáneo, una denominada “no coincidente”, que contiene todas las grabaciones, y otra denominada “coincidente”, que excluye los audios registrados con celulares.

Según Mesaros [56], en colaboración con Heittola, la base de datos TUT-2018 fue la más extensa al momento de su creación y la única con grabaciones de escenas acústicas de varias ciudades en distintos países europeos. Si bien destacan otras bases de datos, los autores explican que de éstas sólo LITIS-ROUEN [57] tiene un tamaño adecuado para utilizar técnicas contemporáneas de aprendizaje mediante computadoras.

Esta última base de datos contiene audios grabados con un teléfono móvil que los mismos autores califican como “realísticamente imperfecto”. También advierten sobre la ambigüedad de ciertas clases, como por ejemplo *calle peatonal*, que por momentos se asemeja a una *calle con tránsito*, aunque esto es común en la mayoría de las bases de datos. En la Tabla 3, se especifican las clases presentes en las bases de datos más importantes.

Tabla 3: Clases de las principales bases de datos para clasificación de escenas acústicas.

LITIS ROUEN			TUT - 2017			TUT - 2018		
N	Nombre	Minutos	N	Nombre	Minutos	N	Nombre	Minutos
1	Avión	12	1	Automóvil	52	1	Aeropuerto	144
2	Autobús	96	2	Café	52	2	Autobús	144
3	Calle	72	3	Autobús	52	3	Metro	145
4	Café	60	4	Centro urbano	52	4	Est. de Metro	144
5	Auto	122	5	Camino forestal	52	5	Parque	144
6	Sala	44	6	Tienda	52	6	Plaza Pública	144
7	Terminal	135	7	Casa	52	7	C. Comercial	145
8	Juego de Niños	73	8	Playa	52	8	Calle peatonal	144
9	Mercado	138	9	Biblioteca	52	9	Calle	144
10	Metro (París)	97	10	Est. de Metro	52	10	Tranvía	145
11	Metro (Rouen)	97	11	Oficina	52			
12	Sala de Pool	78	12	Residencial	52			
13	Calle tranquila	45	13	Tren	52			
14	Restaurante	67	14	Tranvía	52			
15	Calle peatonal	61	15	Parque Urbano	52			
16	Tienda	102						
17	Tren	82						
18	Tren (TGV)	74						
19	Est. de Metro	63						

3.2 REPRESENTACIÓN DE SEÑALES DIGITALES

3.2.1 Representación del sonido

El sonido puede representarse digitalmente a través de los procesos de muestreo, cuantificación y codificación de una señal eléctrica, generada a partir de un transductor acústico. El proceso de muestreo implica tomar muestras de una señal analógica en tiempo discreto, a una frecuencia f_s que es directamente proporcional a la frecuencia máxima (f_{MAX}) de interés de la señal, siendo: $f_s > 2 f_{MAX}$ impuesta por el Teorema de Nyquist [58]. Cuando se trata con sonidos de rango audible (20 Hz - 20 kHz), esta frecuencia es por convención 44.1 kHz, 48 kHz o múltiplos.

Por otro lado, el proceso de cuantificación implica discretizar la amplitud de la señal, donde el número de dígitos binarios requeridos para la representación de una muestra es logarítmicamente proporcional a la cantidad de niveles. Finalmente se codifica la señal, generalmente a través de una *modulación por pulsos codificados* (PCM), como PCM-16 o PCM-FLOAT [59].

La digitalización en sí misma ya es una representación válida de la señal: un modelo acústico podría tener como vector de entrada a la sucesión de muestras discretas cuantificadas de audio. Sin embargo, este sería de muy alta dimensión, proporcional a f_s y a la duración. Ahora bien, la señal puede representarse con una menor dimensionalidad a través de la extracción de descriptores, estratégicamente elegidos en función del modelo y la aplicación. Un ejemplo podría ser tomar la media cuadrática (RMS) de la señal.

Aun cuando no se reduzca la dimensionalidad, la parametrización de señales puede ser útil para simplificar el análisis y procesos de audio, como es el caso de la transformada de Fourier discreta (DFT) [60], la cual consiste en representar una señal $x[n]$, de duración finita N , en un dominio frecuencial. La intuición del proceso supone la proyección de la señal en una base de exponenciales complejas parametrizadas con una fase y magnitud, obteniendo como resultado N componentes $X[k]$ en el plano complejo, siguiendo la Ecuación 1.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn} \quad (1)$$

Al tomar el valor absoluto de los $X[k]$ se obtiene la magnitud de cada componente de frecuencia de la señal, tanto positivas como negativas, que son simétricas entre sí. Estas últimas carecen de sentido físico, de modo que para obtener el espectro de la señal se considera sólo las primeras $N/2 + 1$ componentes (incluyendo el cero), se duplican las amplitudes y se elevan al cuadrado, recuperando la energía original. Debido a que para el cálculo de una DFT se requieren N^2 operaciones, con el fin de minimizar el costo de cómputo, se suele utilizar en su lugar la *transformada rápida de Fourier* (FFT), la cual consiste en la descomposición y reagrupamiento de la transformada en operaciones más simples, disminuyendo la complejidad computacional a $N \log(N)$ operaciones. [61]

Cuando interesa la variación del espectro en el tiempo, se confeccionan espectrogramas a través de la *transformada de Fourier de tiempo reducido* (STFT), la cual consiste en calcular la DFT sobre el audio en ventanas $w[n]$ simétricas de longitud N . Una vez que se obtiene la transformada, se desplaza una cantidad de muestras H (en inglés, “hop size”) y se repite el cálculo hasta agotar las muestras disponibles, como se muestra en la Figura 2.

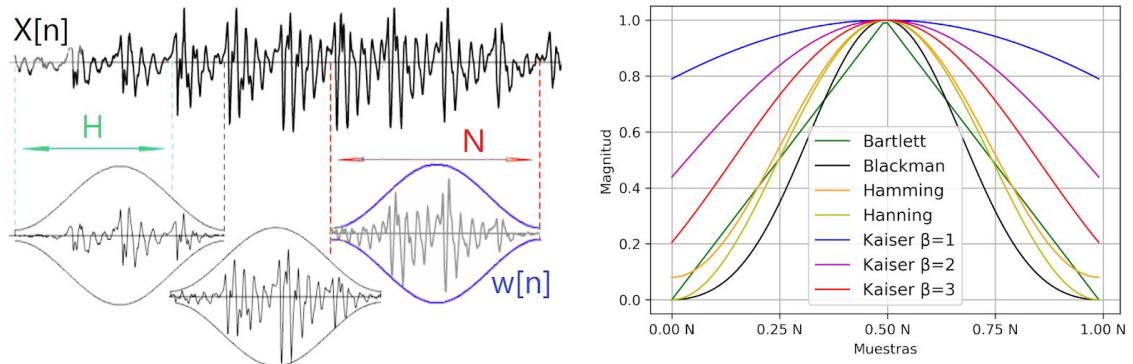


Figura 2: (a) Proceso de ventaneo de una señal. (b) Ejemplos de ventanas típicas.

Se suele utilizar $H < N$ para solapar las ventanas. A mayor N , mayor resolución en frecuencia, aunque es penalizada con un mayor tiempo de análisis, dado por N/f_s , lo que implica una menor resolución temporal [62]. En general, si se tiene un audio de longitud N_T , el espectrograma resultante tendrá N_F cuadros temporales, según la Ecuación 2.

$$N_F = \lfloor \frac{N_T - N}{H} \rfloor + 1 \quad (2)$$

3.2.2 Descriptores de audio

El trabajo de Mitrovic [63] introduce una síntesis de la taxonomía para las técnicas de extracción de los descriptores de audio más relevantes y recientes, aplicados para escenas acústicas (ver Tabla I.1 en el ANEXO I). Según esta taxonomía, los descriptores de audio se dividen en dos grandes grupos: los físicos y los perceptuales. Los primeros refieren a descriptores propios del análisis de la señal, mientras que los segundos describen las propiedades generales del audio basándose en la percepción humana, donde intervienen procesos psicoacústicos. Al mismo tiempo, se pueden clasificar según su dominio: temporal, frecuencial, etc; donde para cada uno existen variantes físicas y perceptuales. Dentro del dominio físico-temporal se destaca la tasa de cruce por cero (ZCR) [64], mientras que en el dominio físico-frecuencial se cuenta con el ancho de banda espectral (SBW) [65] y el *centroide espectral* (SC) [66], los cuales dependen de la frecuencia central f_{CENTER} asociada a cada bin “k”, es decir, a las frecuencias discretas resultantes de la transformación. En este grupo también se encuentra la *planicidad espectral* (SF) [67], el *rodamiento espectral* (SRO) [68] y el *contraste espectral* (SCO) [69]. Las expresiones matemáticas para estos descriptores de escenas acústicas se muestran en la Tabla 4.

Tabla 4: Ecuaciones de descriptores según aparecen en la literatura.

Razón de Cruce por Cero (Zero Crossing Rate)	$ZCR = \frac{1}{2N} \sum_{n=1}^N sign(x[n]) - sign(x[n-1]) $
Centroide Espectral (Spectral Centroid)	$SC = \frac{\sum_{k=0}^{K-1} X[k] f_{center}[k]}{\sum_{k=0}^{K-1} X[k]}$
Ancho de Banda Espectral (Spectral Bandwidth)	$SBW = \sqrt{\sum_{k=0}^K X[k] (f_{center}[k] - SC)^2}$
Planicidad Espectral (Spectral Flatness)	$SF = \frac{\exp\left(\frac{1}{K} \sum_{k=0}^{K-1} \ln X[k]\right)}{\frac{1}{K} \sum_{k=0}^{K-1} X[k]}$
Rodamiento Espectral (Spectral Roll-off Point)	$SRO = f_k \ni \sum_{k=0}^k X[k] \geq 0.85 \sum_{k=0}^{N-1} X[k]$
Contraste Espectral (Spectral Contrast)	$SCO_k = \log\left(\frac{1}{N} \sum_{i=1}^N x_{k,i}\right) - \log\left(\frac{1}{N} \sum_{i=1}^N x_{k,N-i+1}\right)$

Los descriptores perceptuales son diseñados a partir de estudios sobre la percepción humana, entre los que se destacan las mediciones subjetivas realizadas por Stevens [70] sobre frecuencias perceptualmente equidistantes en el año 1937. En este experimento, los sujetos de estudio debían señalar la frecuencia que percibían como la mitad de una frecuencia de referencia determinada, obteniendo la escala Mel, que lleva este nombre en referencia a “melodía”, al estar basada en comparaciones de tonos musicales. La escala obtenida se comporta de forma lineal para bajas frecuencias, donde el sistema auditivo humano puede discernir mejor las pequeñas variaciones, pero comprime el rango para altas frecuencias. Debido a los múltiples métodos experimentales, existen varias ecuaciones para aproximar hercios a mels, aunque actualmente los investigadores optan por la implementación de Slaney [71], en la que se genera un vector f con N_{MEL} componentes equi-espaciadas entre cero y f_{lim} , obtenido a partir de la frecuencia máxima de interés f_{max} , como se muestra en la ecuación.

$$f_{lim} = 15 + 27 \cdot \log_{6.4} \left(\frac{f_{max}}{1000} \right) \quad (3)$$

Luego, escalando el vector f , se obtienen las componentes f_{MEL} de la escala Mel.

$$\begin{cases} f_{mel} = \frac{200}{3} f & f < 15 \\ f_{mel} = 1000 \cdot 6.4^{\frac{f-15}{27}} & f \geq 15 \end{cases} \quad (4)$$

Esto implica una función partida, lineal para frecuencias menores a 1 KHz y exponencial inversa para frecuencias mayores a 1 KHz, como se muestra en la Figura 3, utilizando en este caso 128 componentes Mel.

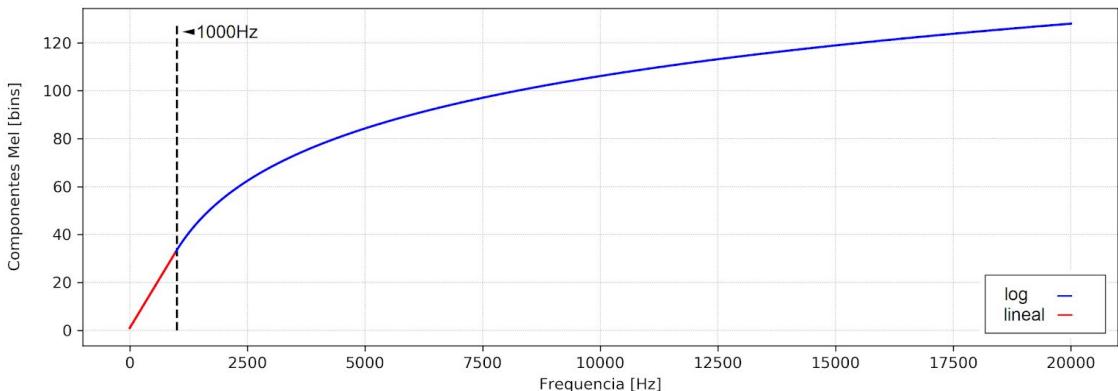


Figura 3: Escala Mel Slaney de 128 componentes con rango de frecuencia 0-20 KHz.

A partir de esta escala, es posible construir un banco de filtros triangulares, uno por cada componente Mel, de modo que los cruces por cero de cada filtro triangular coincidan con los picos de los filtros triangulares adyacentes, exemplificado en la Figura 4 para los mismos 128 componentes, obtenido con la biblioteca “*Librosa*”. Este banco de filtros permite realizar un escalamiento subjetivo de un espectro de potencia lineal obtenido como producto de aplicar una FFT. Este nuevo espectro de escala Mel suele ser utilizado en gran parte de los trabajos relacionados con el reconocimiento de sonidos, como descriptor perceptual.

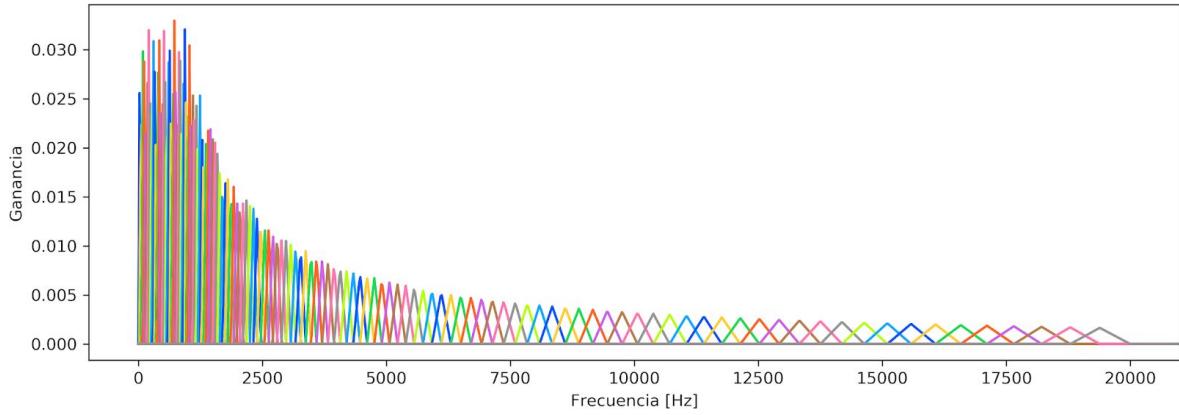


Figura 4: Banco de filtros triangulares Mel Slaney para 128 componentes.

Debido a que sus componentes están altamente correlacionadas, al aplicar una *transformada de coseno discreta* (DCT), se produce una descorrelación del espectro de escala Mel, obteniéndose los *coeficientes cepstrales de frecuencia de Mel* (MFCC), pertenecientes al dominio cepstral. Se pueden extraer tantos coeficientes cepstrales como filtros triangulares tenga un espectro en escala Mel. El *cepstrum de frecuencias Mel* (MFC), cuyo desarrollo es típicamente atribuido a Mermelstein [72], se intuye como una representación del espectro de potencia del sonido en el corto plazo. Los coeficientes son las amplitudes resultantes del espectro, que colectivamente componen al cepstrum [73]. Similarmente, se suelen tomar descriptores dinámicos, conocidos como coeficientes Δ MFCC y $\Delta\Delta$ MFCC, que corresponden a las diferencias de primer y segundo orden entre cuadros temporales. Si bien este descriptor se origina en los procesos de voz, actualmente se lo utiliza en clasificadores de todo tipo de sonidos, incluyendo los sonidos ambientales.

3.3 APRENDIZAJE AUTOMÁTICO

3.3.1 Introducción

El aprendizaje automático, mejor conocido en inglés como *machine learning* (ML), es un campo donde intervienen las ciencias de la computación, la estadística y la inteligencia artificial. Se trata del estudio de algoritmos y modelos estadísticos basados en el reconocimiento de patrones e inferencia sobre datos de muestra, con gran aplicación en tareas de clasificación y regresión [74]. Los avances en el campo han crecido en los últimos años debido al acceso abierto a bases de datos masivas, plataformas de programación libre, computadoras con mayor capacidad de cómputo y, especialmente, por el interés de múltiples sectores de desarrollo.

Las etapas generales del desarrollo de un modelo con ML pueden describirse como:

1. Obtención de base de datos
2. Preprocesamiento de base de datos
3. Elección del modelo
4. Entrenamiento del modelo
5. Evaluación del modelo

Inicialmente, se debe contar con una base de datos representativa y de gran tamaño, obtenida ya sea por medios propios o por terceros. En la etapa de preprocesamiento, se decide cuáles serán las variables de entrada de interés, descartando las de menor utilidad. La elección del modelo y de su estructura interna deben contemplar la representación de los elementos de entrada y de salida, así como la complejidad de la tarea.

El entrenamiento del modelo es un proceso automatizado, en el que el modelo ajusta sus parámetros internos en función de la base de datos de entrada. En este proceso, se experimenta con diferentes configuraciones de los hiper-parámetros, es decir, parámetros fijos que no se entranan, pero que se modifican previo al entrenamiento para mejorar su rendimiento. Finalmente, la evaluación del modelo consiste en verificar la capacidad de generalización del modelo al utilizar nueva información de entrada, exclusivamente reservada para esta etapa, de la misma forma que un profesor no revela los contenidos de su examen durante una de sus clases.

3.3.2 Preprocesamiento de datos

El rendimiento de los modelos depende principalmente del tamaño y la calidad de la base de datos utilizada para su entrenamiento. En problemas de clasificación, por ejemplo, usualmente se tiene una distribución desigual de información según la clase, que podría influir en la etapa de aprendizaje del modelo, imprimiendo un sesgo en las predicciones a favor de las clases con mayor cantidad de muestras, en la medida que generaliza que de esta manera obtiene una alta proporción de aciertos. Al tratar con una distribución de datos desequilibrada, se podrían obtener buenos resultados durante el entrenamiento y pobres durante la evaluación; por esta razón, se intenta utilizar bases de datos balanceadas, homogéneamente distribuidas.

Las técnicas de preprocesamiento de datos permiten intervenir la información que se utilizará para el entrenamiento de un modelo. El proceso implica analizar la base de datos para determinar si contiene información redundante, irrelevante, ruidosa o desconfiable. A partir de ello, se pueden aplicar técnicas de filtrado, de-duplicación, selección, normalización, transformación, entre otras.

Es posible incrementar el tamaño aparente de una base de datos a través de técnicas de aumento de datos (DA), que consiste en técnicas de estimación, extrapolación, análisis estadísticos y otros procesos para generar datos nuevos [75]. En el caso del sonido, se suelen sintetizar nuevas versiones de audios, introduciendo variaciones sutiles mediante procesos de distorsión, filtrado, enmascaramiento y agregado de ruido, lo que tiene validez en la medida que la nueva información siga perteneciendo a su clase original y sea representativa.

Como producto del preprocesamiento se obtienen tres conjuntos de datos de entrada: uno de entrenamiento (training-set), utilizado por el modelo recurrentemente durante el ajuste de sus parámetros internos; uno de validación interna (validation-set), utilizado para medir la performance y el grado de sobreajuste del modelo al entrenarse; y por último, uno de evaluación (evaluation-set), utilizado para determinar la capacidad de generalización del modelo entrenado y su validez externa. La distribución de datos entre los conjuntos suele variar según los autores, aunque generalmente se le asigna mayor proporción al training-set [76].

3.3.3 Modelos

Los modelos de aprendizaje automático se basan en el uso de algoritmos para encontrar patrones en la información y poder llevar a cabo tareas de clasificación y regresión. Éstos deben ser capaces de aproximar el comportamiento del sistema, que se valida utilizando información empírica, obtenida del mundo real.

En todos los casos, se diferencian tres tipos de aprendizaje. En primer lugar, el aprendizaje supervisado, que consiste en utilizar bases de datos previamente etiquetadas. Estos modelos ajustan sus parámetros internos para minimizar el error de predicción. En el caso de que la variable de salida buscada sea continua, se trata de una tarea de regresión, mientras que, si las salidas son categorías discretas, el modelo constituye un clasificador, cuyo entrenamiento se refleja en el esquema sinóptico de la Figura 5.

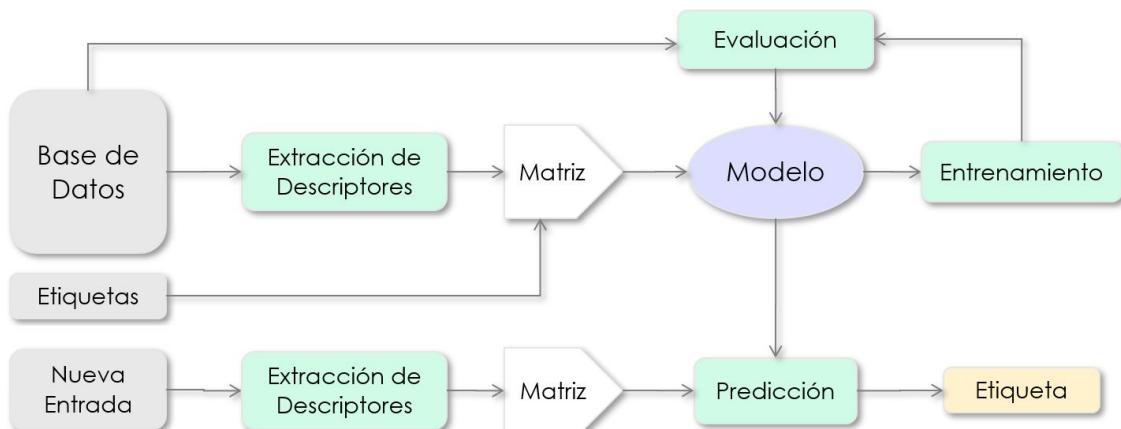


Figura 5: Diagrama de flujo de aprendizaje supervisado para un modelo clasificador.

Luego se encuentra el aprendizaje no-supervisado, donde se utilizan bases de datos no etiquetadas, por lo que el sistema se debe valer del reconocimiento de patrones previamente desconocidos en la estructura de los datos. Típicamente consiste en técnicas de agrupamiento (*clustering*) y auto-codificadores.

Por último, se encuentra el aprendizaje por refuerzo, donde se penalizan o castigan las acciones de un agente virtual que ejecuta acciones dentro de un entorno predeterminado, con el objetivo de maximizar la recompensa y minimizar los castigos.

Los modelos generalmente se suelen diferenciar por su aplicación: clasificación o regresión. En el primer caso, la salida del modelo, representa la probabilidad de pertenencia de la entrada a una clase predeterminada, generalmente codificada de forma binaria (*one-hot*). En contraste, un modelo de regresión arroja una predicción numérica continua que representa la estimación de alguna variable en función de la entrada.

Al basarse en el reconocimiento de patrones de datos de entrenamiento, los modelos de ML suelen presentar dificultades dependiendo de las características de la información. Ciertos modelos pueden lograr un buen desempeño partiendo de una base de datos pequeña para tareas relativamente sencillas, sin embargo, para tareas de mayor complejidad, los modelos requieren de bases de datos de gran tamaño, las cuales no siempre están disponibles.

Por otro lado, se encuentra el problema de la dimensionalidad: si la información de entrada tiene muy alta dimensión, el volumen del espacio de descriptores aumenta y los datos de entrenamiento se vuelven virtualmente escasos, perjudicando el ajuste de los algoritmos estadísticos. La cantidad de datos necesarios crece exponencialmente junto a la dimensionalidad [77]. Este problema es generalmente mitigado a través de técnicas de reducción de dimensionalidad según el caso.

Otro problema frecuente es el denominado sobreajuste, es decir, cuando existen variaciones no representativas en la información de entrada, se reconocen falsamente como patrones importantes en el sistema. Debido a que el modelo siempre se ajusta en función de los datos de entrenamiento, si su complejidad interna lo permite, éste puede adaptarse al ruido de los datos perjudicando su capacidad de generalización.

Puede hallarse una gran variedad de clasificadores adecuados para diferentes aplicaciones, entre ellos, los dedicados a la clasificación de números, texto, secuencias, imágenes, video y audio, entre los que se destacan: los de agrupamiento [78], los árboles de decisiones, las máquinas de vectores de soporte (SVM) [79] y los K-Vecinos más Cercanos (KNN) [80]. Sin embargo, con los avances informáticos, las redes neuronales artificiales (ANN) actualmente son las más utilizadas para problemas de clasificación categórica, debido a que permiten obtener mejores resultados en general, dependiendo de las características de los datos y de la aplicación del modelo.

3.3.4 Redes neuronales artificiales profundas

Las redes neuronales artificiales están inspiradas en las redes neuronales biológicas, formadas por células llamadas neuronas, que reciben señales de entrada mediante conexiones sinápticas a través de axones [81]. Cada neurona realiza un proceso muy simple en base a sus entradas y su función de activación. El modelo se organiza en capas, estableciendo una estructura jerárquica, como se muestra en la Figura 6.

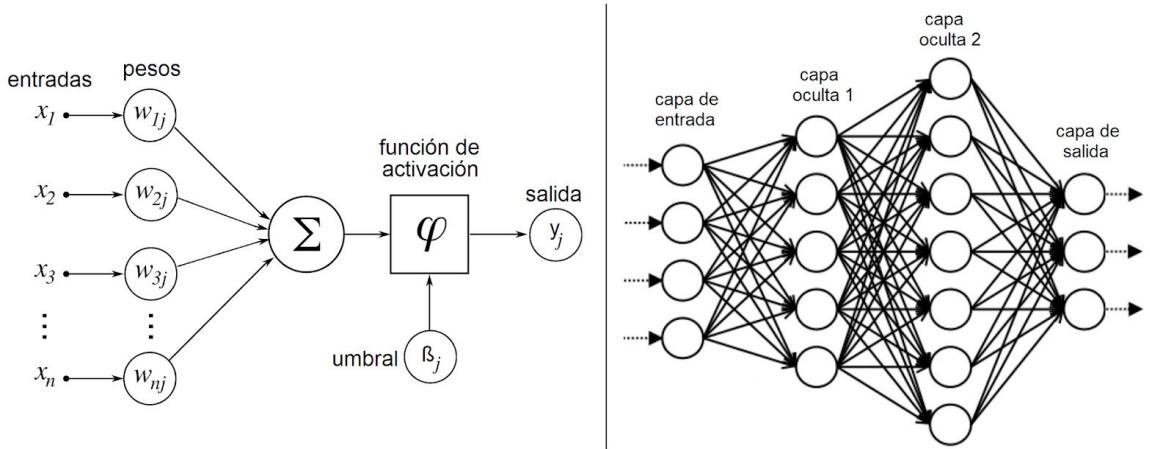


Figura 6: (a): Esquema de neurona. (b): Ejemplo de estructura de capas. Adaptado de [82].

En una capa densamente conectada, cada neurona tiene como entrada a todas las de la capa anterior, con un peso (*weight*) asociado en cada conexión, de forma que permite dar mayor importancia a ciertas conexiones por sobre otras. Por otro lado, existe un parámetro adicional llamado umbral (*bias*), que consiste en una constante que controla la activación.

Matemáticamente, se puede pensar a la salida escalar y de una neurona como el producto interno de un tensor de entrada \mathbf{X} con una matriz de pesos \mathbf{W} sumado a una matriz umbral $\boldsymbol{\beta}$, al que luego se le aplica una función de activación ϕ , como se muestra en la ecuación (5). De aquí en adelante se referirá como “pre-activación” al argumento de la función de activación.

$$y = \phi(W \cdot X + \beta) \quad (5)$$

Partiendo de la idea original del perceptrón desarrollado a mitad del siglo XX por Frank Rosenblatt [83], una neurona se activa si la pre-activación es positiva, o bien, se desactiva si es negativa. La activación implica una salida binaria de “1”, mientras que la desactivación implica un “0”. Esto constituye lo que se denomina “función de activación”, que, en el caso del perceptrón, se trata de una función escalón.

A medida que avanzó el tiempo, aparecieron funciones de activación con salidas no binarias, sino proporcionales a la magnitud de la pre-activación, con números reales (R) entre $-\infty$ y $+\infty$. No obstante, trabajar con un rango tan extenso puede resultar problemático, por lo que es preferible reducirlo a un rango acotado a través de funciones matemáticas como la función *sigmoide*. Ha de tenerse en cuenta que durante el entrenamiento aparecen las derivadas de estas funciones en la expresión que optimiza los pesos, de forma que, si la derivada de la función de activación es cero, los pesos dejan de ajustarse. Por esta razón generalmente se utiliza la función de activación Re-LU (*rectified linear units*), que consiste en una rampa de pendiente creciente constante para valores positivos, y nula para los valores negativos, tal como se muestra en la Figura 7.

En el contexto de un clasificador en el que las clases son mutuamente excluyentes, las neuronas de la última capa corresponden a la predicción de clase, que debe ser una distribución de probabilidad, es decir, sus activaciones deben sumar 1. Esto se logra utilizando una capa especial llamada *softmax*, que se encarga de normalizar la suma.

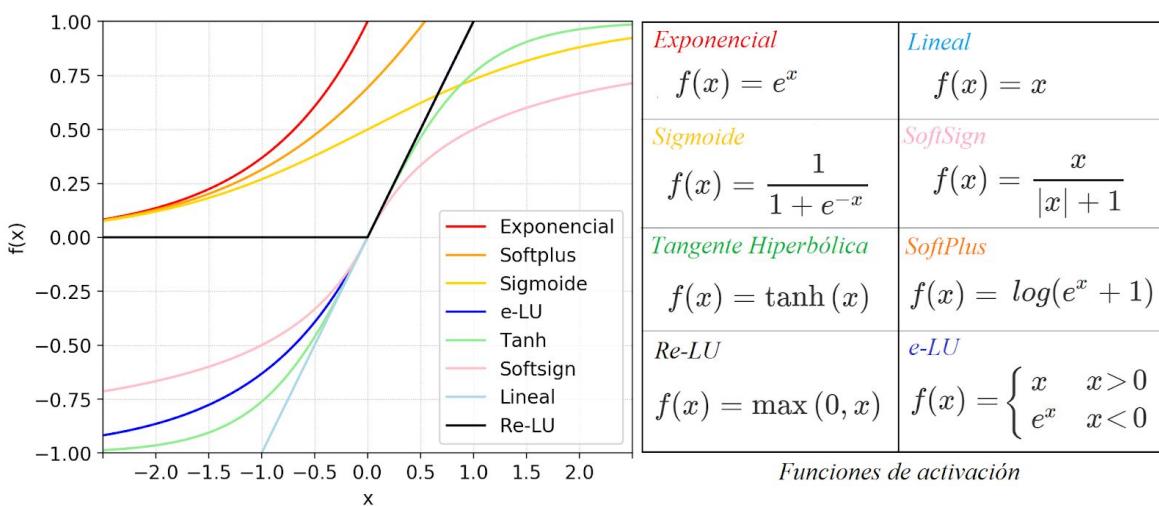


Figura 7: Funciones de activación comúnmente utilizadas para redes neuronales artificiales.

3.3.5 Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) fueron popularizadas luego de su éxito en tareas de clasificación de imágenes, como en los trabajos de Krizhevsky [84] y Simonyan [85]. Si bien surgen para modelar procesos con dominio de imagen, hoy han alcanzado el estado del arte en el dominio de sonido, y en particular para escenas acústicas, como en el trabajo de Han [86] sobre representaciones binaurales.

Estas redes admiten tensores de entrada con múltiples dimensiones, aunque en el campo del procesamiento de imagen, típicamente son tres: $I_H \times I_W \times I_D$ (alto, ancho y cantidad de canales). Los filtros de una capa convolucional constituyen los parámetros ajustables de una CNN, los cuales se entrena a través del aprendizaje automático para realizar procesos complejos. El proceso de convolución tiene el efecto de transformar la entrada de manera tal que ciertas características, determinadas por el filtro, se destaque. Los filtros en la capa convolucional se superponen a una región de la entrada donde se calcula una convolución, luego se los desplaza una cantidad de pasos (*stride*) y se repite la operación hasta cubrir la totalidad de la entrada, almacenando los resultados en un nuevo tensor, que constituye la salida. En el caso que los filtros queden fuera de la extensión de la imagen, se descarta el sector en conflicto y la salida sufre una reducción de dimensión dada por: $(I_H - f_H + 1) \times (I_W - f_W + 1)$. Para evitar una reducción de dimensión tras el proceso de convolución se suelen utilizar técnicas de relleno (padding), las cuales no afectan la aplicación del filtro convolucional [87].

Tras una capa convolucional, se suele aplicar una capa de agrupamiento (*pooling*). Esta capa reduce la dimensión de entrada a través de un agrupamiento de tamaño $P_H \times P_W$ predefinido, que se desplaza por la imagen de manera similar a los filtros de una capa convolucional. En este caso, el proceso consiste en ir extrayendo el promedio de los grupos (*average-pooling*), o bien el máximo (*max-pooling*). La capa convolucional, en conjunto a la capa de agrupamiento, componen un bloque constructivo que se puede utilizar en cadena, intercalado con capas de activación, generalmente Re-LU.

Las operaciones de convolución matricial y agrupamiento se ejemplifican conceptualmente a través de la Figura 8.

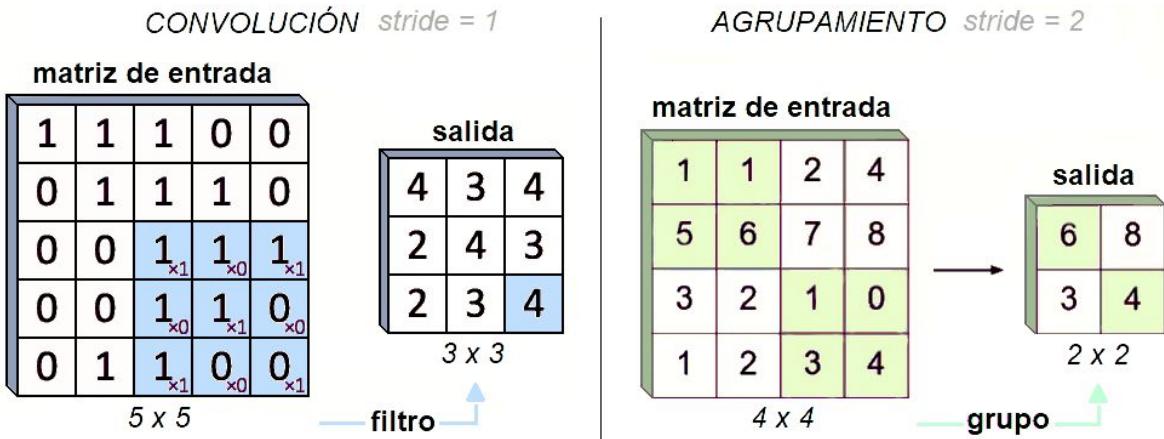


Figura 8: Ejemplos de las operaciones de convolución 2D y agrupamiento de máximos.

La cantidad de filtros en cada capa, el tamaño de cada filtro, la cantidad de pasos y el tipo de relleno son los hiper-parámetros de las CNN, los cuales se modifican para optimizar el proceso. Por ejemplo, si se tiene como entrada un spectrograma de dimensión $N_{\text{bins}} \times N_{\text{cuadros}}$ es posible extraer características temporales aplicando filtros de dimensión $1 \times N_{\text{cuadros}}$ o bien características frecuenciales, utilizando filtros de $N_{\text{bins}} \times 1$ [88].

Tras una serie de capas convolucionales y de agrupamiento, se suele utilizar una capa de achatamiento que redimensiona los tensores de salida de una CNN a un tensor unidimensional, compatible con capas propias de una red completamente conectada, que, en el caso de los modelos clasificadores, culminan en una *softmax*, en donde las neuronas de salida reflejan la predicción de la red, como se ilustra en la Figura 9.

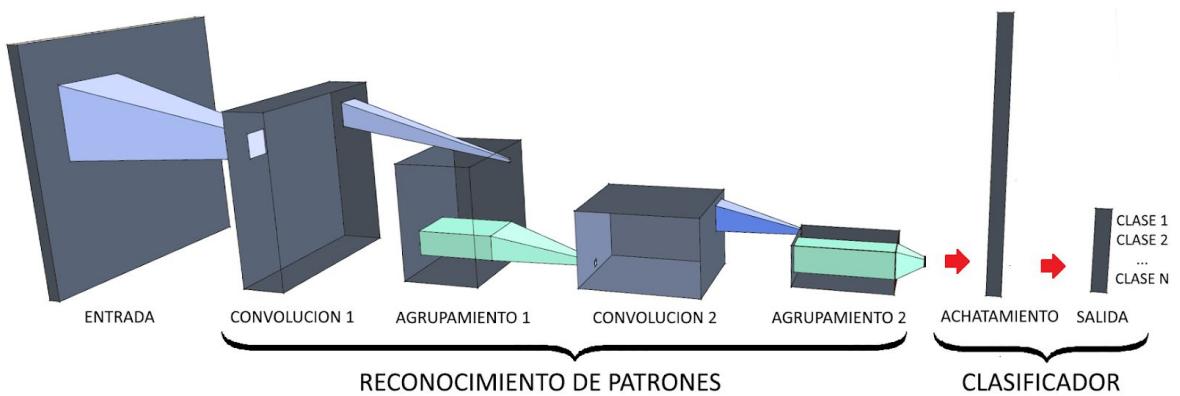


Figura 9: Esquema ilustrativo de una CNN para clasificación. Adaptado de [89].

3.3.6 Entrenamiento de redes neuronales

El entrenamiento supervisado de un clasificador consiste en ingresar tanto datos de entrenamiento como etiquetas, es decir, información sobre la clase correcta, representada mediante índices o bien implementando una codificación *one-hot*, es decir, un vector binario de longitud equivalente a la cantidad total de clases. Durante el entrenamiento de una red neuronal se ajustan automáticamente los parámetros que se pueden entrenar (pesos y umbrales) de modo que el modelo minimiza su error de predicción sobre los datos de entrenamiento.

Los parámetros internos se inicializan a valores aleatorios, lo que implica que ante cualquier entrada, las N neuronas de salida también serán aleatorias. La diferencia entre los valores correctos (y_{ij}) y los obtenidos (\hat{y}_{ij}) para las M clases se denomina “error” del modelo, que es calculado a partir de una función de costo (L), la cual puede ser entropía cruzada de Bernoulli para clasificadores binarios [90], o bien entropía cruzada categórica para clasificadores multiclase, la cual se corresponde con la Ecuación 6.

$$L(y, \hat{y}) = - \sum_{j=0}^M \sum_{i=0}^N y_{ij} \cdot \log(\hat{y}_{ij}) \quad (6)$$

La minimización del error es un problema de optimización, el cual se puede resolver utilizando un algoritmo iterativo conocido como descenso por gradiente. El algoritmo consiste en evaluar la dirección del gradiente, es decir, hallar el camino hacia el mínimo local de la función. A través del proceso propuesto por Rumelhart [91] denominado “propagación hacia atrás”, (*backpropagation*), es posible determinar cuáles son los parámetros de la capa anterior que se deben ajustar, en qué sentido y en qué proporción, capa por capa, desde la última hacia la primera. Debido a que el cálculo del gradiente es un proceso computacionalmente costoso, éste se aplica iterativamente agrupando las muestras de entrenamiento en lotes (*batches*) de tamaño predeterminado (*batch size*).

Un tamaño de lote elevado disminuye la frecuencia con la que se calcula el error y se ajustan los pesos, lo que resulta en un menor costo computacional. Eventualmente, se agotan las muestras de entrenamiento, habiendo transcurrido una denominada época de entrenamiento (*epoch*). Para continuar ajustando el modelo es posible reutilizar las muestras un número predeterminado de épocas, alterando el orden en cada iteración. No obstante, el error de entrenamiento no decrece indefinidamente y puede suceder que a partir de determinada época el error de validación comience a aumentar.

Entre los parámetros de aprendizaje más importantes se encuentra la tasa de aprendizaje, que influye en la magnitud del ajuste entre cada ciclo de propagación, pudiendo variar de época en época: un valor demasiado elevado saltaría los mínimos locales, mientras que uno muy reducido podría estancarse en un mínimo local indeseado.

Con el fin de variar automáticamente los parámetros de aprendizaje durante el entrenamiento, se utilizan algoritmos de optimización, como el de ADAM (*adaptive moment estimation*), el cual es una extensión del algoritmo de descenso por gradiente estocástico que suele ser aplicado para resolver problemas donde se utilizan redes profundas.

Durante el entrenamiento de una ANN, la distribución de entrada de cada capa varía, lo que ralentiza el ajuste. Este problema se reduce tras la introducción de técnicas de normalización de lote, aplicadas en los mini-lotes durante el entrenamiento, corrigiendo el desplazamiento de covarianza interna de la capa de entrada [92].

Cuantos más grados de libertad tenga un modelo, más susceptible es al sobreajuste, ya que puede acomodar sus parámetros internos de modo que el error sea mínimo, desarrollando extrema dependencia con los datos de entrenamiento. Un sobreajuste implica que el modelo no logra generalizar correctamente los patrones intrínsecos, haciendo que el rendimiento sobre información desconocida resulte pobre. En su fase de sobreajuste, el modelo mejora su rendimiento con datos de entrenamiento, pero empeora con los de validación [93]. Para prevenir el sobreajuste se intenta eliminar la dependencia de ciertas neuronas en otras, en una técnica denominada *dropout*, donde sistemáticamente se ignoran neuronas de forma aleatoria durante el entrenamiento [94].

3.3.7 Evaluación de clasificadores

Una vez que se finaliza el entrenamiento de un clasificador, es el momento de evaluar su rendimiento. La evaluación consiste en obtener el porcentaje de aciertos y errores del modelo, utilizando información de entrada ausente en su etapa de entrenamiento. A partir del análisis de los verdaderos positivos (TP), falsos negativos (FN), verdaderos negativos (TN) y falsos positivos (FP), es posible obtener métricas de rendimiento como son la sensibilidad (*TPR, true positive rate*), la especificidad (*TNR, true negative rate*), la precisión (*PPV, predicted positive values*), el valor predictivo negativo (*NPV*), la exactitud (*ACC, accuracy*), y el valor-F (*F1 Score*), que representa la media armónica entre la sensibilidad y la precisión [95]. Estas métricas se definen a través de las ecuaciones 7 - 12.

$$TPR = \frac{TP}{TP + FN} \quad (7)$$

$$TNR = \frac{TN}{TN + FP} \quad (8)$$

$$PPV = \frac{TP}{TP + FP} \quad (9)$$

$$NPV = \frac{TN}{TN + FN} \quad (10)$$

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

$$F1 = 2 \frac{PPV \cdot TPR}{PPV + TPR} \quad (12)$$

La precisión es la proporción de predicciones verdaderas positivas sobre el total de predicciones positivas, es decir, un modelo con alta precisión presentará un escaso número de falsos positivos. Por otro lado, la sensibilidad es la proporción de predicciones verdaderas positivas sobre el total de positivos de la población, es decir, un modelo con alta sensibilidad presentará un escaso número de falsos negativos. Análogamente, la especificidad es la proporción de predicciones verdaderas negativas sobre el total de negativos de la población, es decir, un modelo con alta especificidad presentará un escaso número de falsos positivos.

Entre estas métricas, el F-valor se destaca para caracterizar de forma simple el rendimiento de los modelos clasificadores. Esto se debe a que se obtiene a partir de dos de las métricas más relevantes (precisión y la sensibilidad).

Una forma de visualizar el rendimiento del modelo es a través de una matriz de confusión, donde las columnas representan las predicciones mientras que las filas representan las etiquetas verdaderas, idealmente formando una matriz diagonal. Para problemas multiclase, se analiza cada clase C_k en función del resto de forma binaria, como se muestra en la Figura 10.

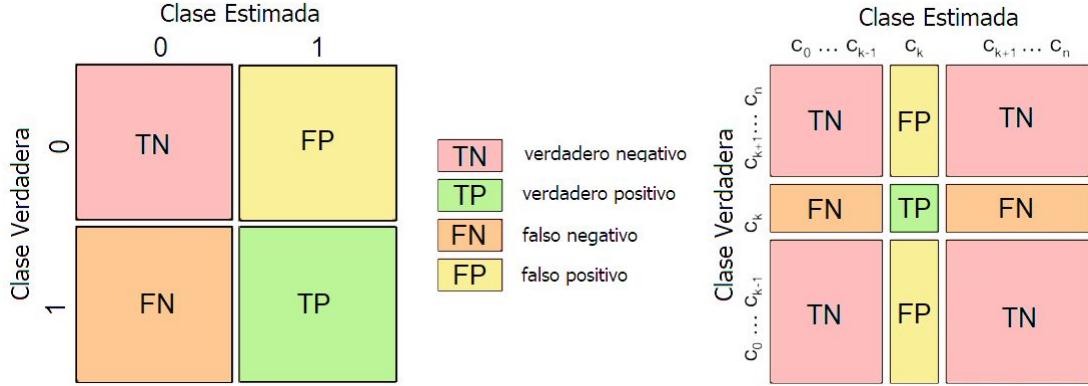


Figura 10: Matriz de confusión para modelo binario (izq) y multiclase (der). Adaptado de [96].

Otra representación que permite visualizar el desempeño del modelo es la curva ROC (por sus siglas en inglés *receiver operating characteristic*), asociada a la separación entre clases binarias, aplicando el método uno-contra-todos para problemas multiclase. El gráfico compara la sensibilidad y el inverso de la especificidad, como se muestra en la Figura 11. El caso óptimo es cuando el área bajo la curva (AUC) tiende a 1, mientras que si el AUC tiende a 0.5, se considera un clasificador azaroso [97]. Para las curvas intermedias se debe tomar un umbral de clasificación priorizando la minimización de FP o bien de FN.

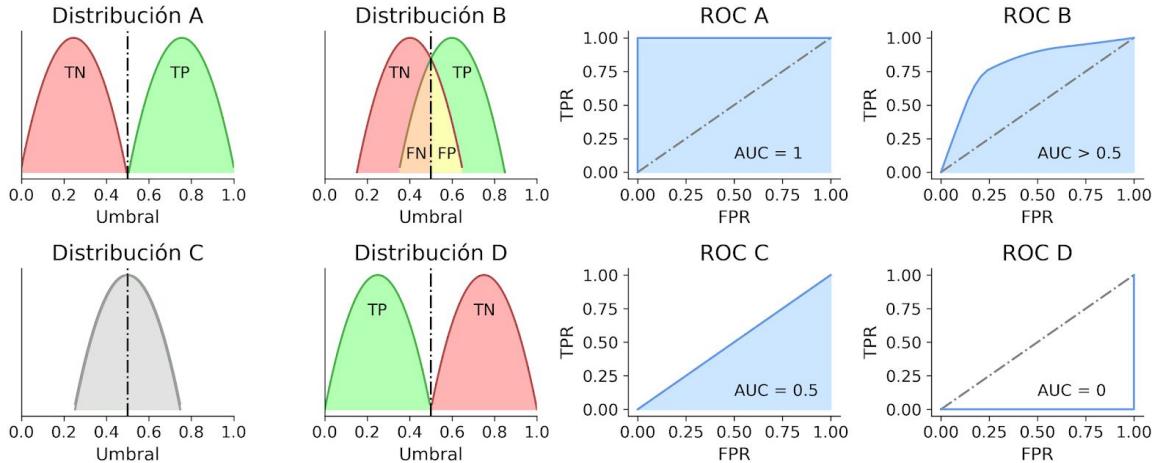


Figura 11: Ejemplos de análisis ROC para diversas distribuciones binarias.

4. METODOLOGÍA

Siguiendo el objetivo planteado en esta tesis, luego de la recopilación de trabajos previos y elaboración del marco teórico, se define una metodología de investigación, la cual atraviesa una serie de desafíos que parten de la adquisición de una base de datos de escenas acústicas, el estudio y extracción de descriptores de sonido significativos, el desarrollo de un clasificador predictivo a través de pruebas preliminares y, finalmente, en la implementación del modelo final en una aplicación móvil no comercial. La metodología utilizada en cada una de las etapas se detalla en esta sección.

4.1 BASES DE DATOS

4.1.1 Selección y procesamiento

Tras la búsqueda exhaustiva de bases de datos de escenas acústicas disponibles, se decide utilizar TUT-2017 y TUT-2018, ya que están parcialmente registradas con dispositivos móviles y son las de mayor tamaño y mejor balance. Si bien en un principio se considera utilizar también LITIS-ROUEN, al analizar su distribución espectral promedio, disminuida en comparación con las TUT (ver Figura I.1 del ANEXO I), se determina descartarla.

Al inspeccionar las muestras, se encuentra que un número significativo de audios presentan ruidos e interferencias, por lo que se realiza una limpieza exhaustiva. Esto se lleva a cabo a partir de la detección manual y análisis de eventos sonoros. Los audios más problemáticos son los que pertenecen a la clase autobús, ya que presentan gran cantidad de golpes de micrófono. En segundo orden, los audios de parques urbanos y caminos forestales contienen muchos fragmentos inutilizables por la presencia de interferencias. Asimismo, presentan una proporción desmedida de sonidos de fuentes y tránsito.

Luego, se encuentran los sonidos considerados impropios dentro de una escena, como es el caso de los trenes, que en la mayoría de los casos presentan ruidos de cubiertos y platos, típicos de trenes europeos, pero que no son representativos de trenes de otras regiones.

Se decide preservar los sonidos de alarmas y señales acústicas presentes en medios de transporte como autobuses, subterráneos y trenes, así como los avisos en las estaciones de metro, debido a que se consideran representativos de la escena, aunque se advierte que su sonido difiere de las alarmas empleadas en regiones fuera de Europa.

Si bien las bases de datos contienen audios parcialmente registrados a través de teléfonos celulares con respuestas en frecuencia características, existe una gran proporción que no; por este motivo y con el fin de homogeneizar los espectros de los audios destinados al entrenamiento, se aplican filtros pasa-altos (HPF), de primer y segundo orden, es decir, con pendientes de 6 y 12 dB por octava, con frecuencias de corte centradas en 200 Hz, virtualmente duplicando la cantidad de datos disponibles y restringiendo el ancho de banda a frecuencias normalmente captadas por la mayoría de los teléfonos móviles. En el caso de las señales estéreo o biaurales, se realiza una conversión a mono.

En la Figura 12 se muestran los espectros de todas las clases promediados globalmente, antes y después de aplicar los procesos de bases de datos, donde se refleja la atenuación en bajas frecuencias.

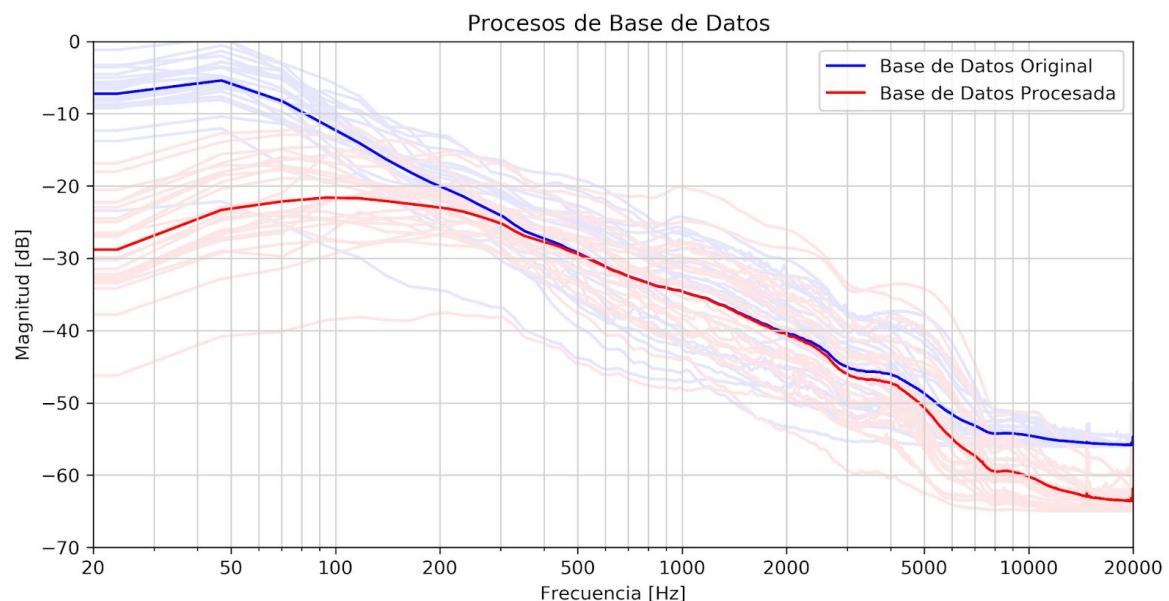


Figura 12: Espectros promediados tras la aplicación de procesos de bases de datos.

4.1.2 Combinaciones

Para la selección y desarrollo del modelo en este trabajo, se recurre a la combinación de clases pertenecientes a distintas bases de datos, con el fin de generar una nueva colección, que abarque mayor variedad de escenas acústicas. El criterio de agrupamiento es reunir las clases con características similares o confundibles, como en este caso ocurre para las clases *casa*, *oficina* y *biblioteca*. Existen clases que son exclusivas de una base de datos, como es el caso de *café*, *centro comercial*, y los transportes *tren*, *automóvil* y *metro*, las cuales no se combinan. Luego están las clases presentes en ambas como *bus*, *estación de metro* y *parque*, que sí se combinan. Se opta por no utilizar los sonidos de *playa*, *tranvía* y *aeropuerto* ya que son escenas poco frecuentes o inexistentes en muchas ciudades. Con el propósito de balancear las muestras combinadas, se lleva a cabo la eliminación aleatoria de audios, hasta emparejar la distribución de muestras por clase.

En el esquema de la Figura 13 se muestra la combinación de clases, generada a partir de las bases TUT-2017 y TUT-2018, utilizada durante los experimentos que se desarrollan en este trabajo. La base de datos resultante se denomina “TUT-COMBI”.

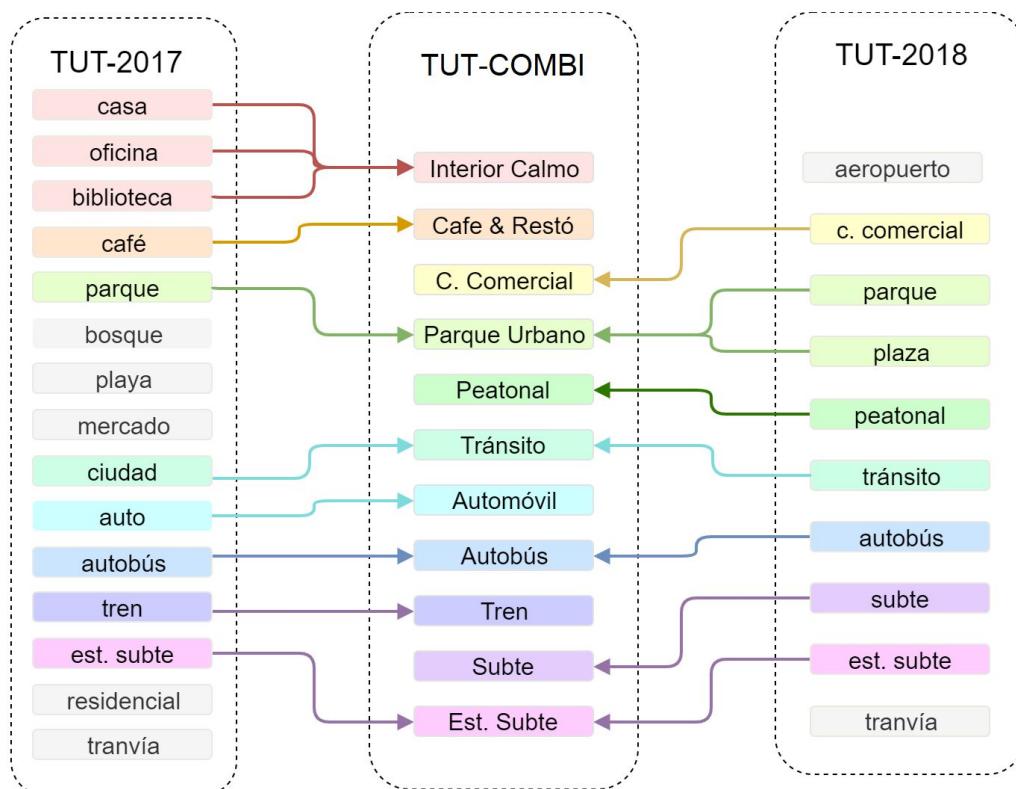


Figura 13: Combinación de bases de datos utilizadas para los experimentos con modelos ANN.

4.1.3 Base de datos de evaluación

Con el fin de evaluar externamente el modelo desarrollado, es necesario utilizar muestras de datos diferentes a las empleadas para el entrenamiento. Las bases de datos TUT-2017 y TUT-2018 cuentan con una partición dedicada a la evaluación, sin embargo, no publican las etiquetas asociadas, de modo que no es posible evaluar los modelos generados con la base de datos combinada. Se opta entonces por generar una base de datos a partir de sonidos registrados en la Ciudad de Buenos Aires (BA-DB), basada en las clases que integra TUT-COMBI, detallada en 4.1.2. En la Tabla 5 se detallan las locaciones y el tiempo de grabación total para cada clase.

Tabla 5: Locaciones de Buenos Aires seleccionadas para registrar las señales de evaluación.

N	Clase Locaciones de Grabación	Duración [mm:ss]
0	Interior Calmo <i>Interior Hogares, Biblioteca UNTREF</i>	20:00
1	Café-Resto <i>Cafe. Pertutti, Rtes. Continental, Tercetas y Cuartetas</i>	20:18
2	Centro Comercial <i>Unicenter, Shop Adidas, Shop JFS, Shop Nike</i>	15:25
3	Parque Urbano <i>Costanera Sur, Pza. Retiro, Pza. San Martín, Pza. Lavalle</i>	15:03
4	Calle Urbana <i>Peat. Florida, Peat. Carabelas</i>	13:47
5	Calle Tránsito <i>Au. Panamericana, Av. 9 Julio, Av. Alem</i>	14:01
6	Autobús <i>Líneas: 21, 59, 71, 152, 161</i>	18:36
7	Automóvil <i>C. Paraná, Av. Santa Fe, Au. Acceso Norte</i>	20:27
8	Tren <i>Líneas: Mitre, San Martín, Urquiza</i>	16:25
9	Metro <i>Líneas: B, D</i>	11:14
10	Est. de Metro <i>Estaciones Congreso, Lacroze, Tribunales</i>	12:19

En cada una de las locaciones se registran entre 10 y 20 minutos de audio en formato *wav*, equivalente a un total de espectrogramas superior a 7000 muestras, utilizando el micrófono de un dispositivo móvil. Los registros de audio se llevan a cabo sosteniendo un celular con una o dos manos, simulando un caso de uso típico, evitando interferir con el campo de captación del micrófono. El dispositivo móvil que se utiliza para las pruebas es el Moto-G4 de Lenovo. Para llevar a cabo los registros de escenas, se utiliza una aplicación² especialmente desarrollada para este trabajo, la cual permite grabar muestras de audio en una resolución de 32 bit por muestra con punto flotante con una frecuencia de muestreo de 48 kHz.

² Código fuente: <https://github.com/lucianodebortoli/SceneClassifier/tree/master/android/Recorder>

4.2 DESCRIPTORES

4.2.1 Método de extracción

En este trabajo se experimenta utilizando descriptores de audio pertinentes para escenas acústicas, recomendados en los trabajos previos. Entre ellos se enumeran: Tasa de Cruce por Cero (ZCR), Frecuencia de Rodamiento espectral (SRO), Centroide espectral (SC), Planicidad espectral (SF), Ancho de banda espectral (SBW), Contraste espectral (SCO), Coeficientes cepstrales de frecuencia Mel (MFCC) y el espectrograma de potencia en escala Mel.

Estos descriptores se pueden obtener utilizando el paquete *librosa*³, disponible para su uso con el lenguaje de programación *Python*. La mayoría de éstos admiten como argumentos parámetros tales como el tamaño de ventanas y de salto, que se configuran en 2048 y 1024, respectivamente. Por otro lado, se define la cantidad de filtros triangulares utilizados para el cómputo de espectros de escala Mel y MFCC, configurados en 128 y 50 respectivamente. En el caso de un espectrograma, la relación entre estos parámetros determina la resolución frecuencial y temporal, así como su duración, tal como se observa en la Figura 14 para señales con frecuencia de muestreo de 48 kHz.

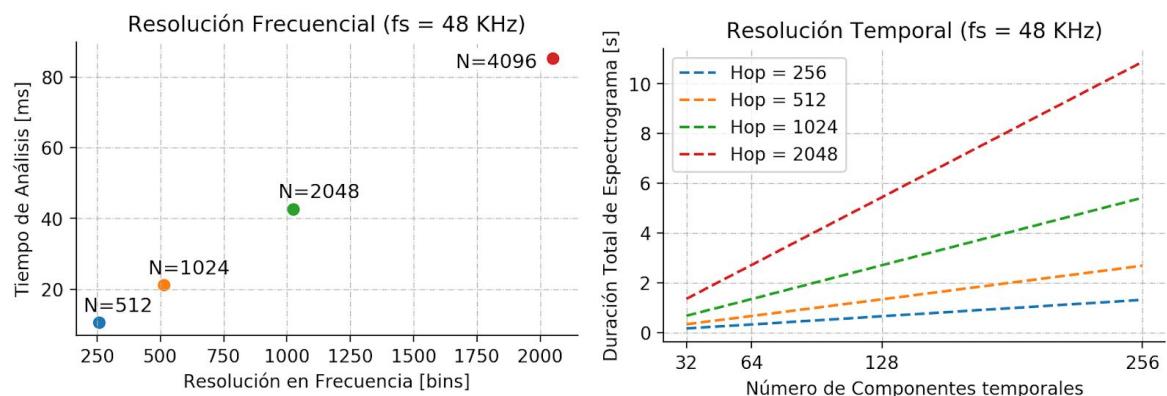


Figura 14: Influencia de N y H en la resolución frecuencial (a) y temporal (b) del espectrograma.

En este caso, se decide utilizar tamaños de FFT de 2048 muestras, con saltos cada 1024, lo que implica una duración total de espectrograma de 2.71 segundos, que es similar al utilizado en trabajos previos [31].

³ Código fuente en: https://librosa.github.io/librosa/_modules/librosa/feature/spectral.html

4.2.2 Selección de atributos

Con el propósito de identificar los descriptores de sonidos más determinantes para la clasificación de escenas acústicas, se utiliza una red denominada “DNN-0”, con dos capas de 50 neuronas, la cual se entrena durante 200 épocas, utilizando lotes de 32 muestras de las bases de datos TUT-2017 y TUT-2018 procesadas. Cada combinación de atributos constituye un nuevo modelo, como se ilustra en la Figura 15. Los descriptores se evalúan en función de los resultados de F-valor obtenidos. En ANEXO II se encuentran los resultados completos.

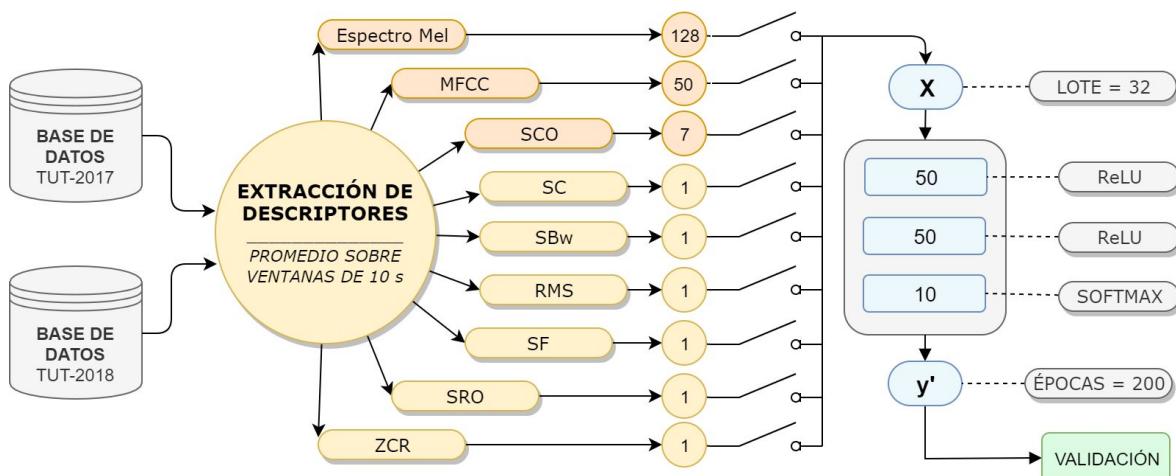


Figura 15: Esquema para la combinación de descriptores de escenas acústicas.

Luego se repiten los experimentos sobre los descriptores destacados, variando la cantidad de capas (entre 1 y 4), y de neuronas (entre 25 y 200), como se muestra en la Figura 16.

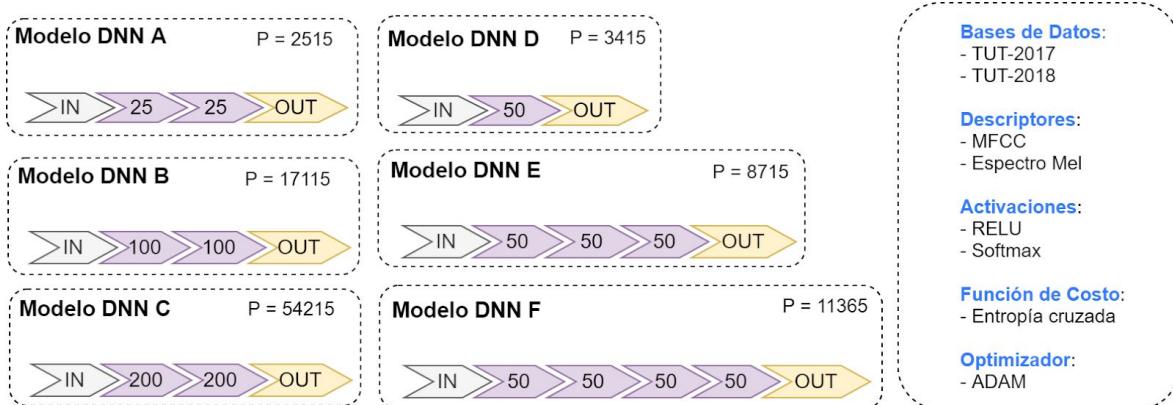


Figura 16: Variaciones de DNN para la selección de descriptores de escenas acústicas.

4.3 MODELO PREDICTIVO

En este trabajo se desarrollan modelos basados exclusivamente en ANN debido a su rendimiento según el estado del arte para tareas de clasificación. Para el desarrollo de estos modelos se utiliza la plataforma abierta *TensorFlow*⁴, la cual cuenta con varias herramientas para el desarrollo de algoritmos de ML. Éstos se construyen como operaciones entre variables, organizadas mediante grafos, utilizando *Python* a modo de *front-end*, aunque muchos procesos se ejecutan en C++ para mejorar el rendimiento de cómputo.

Para este trabajo, se utiliza *Keras*⁵, una biblioteca que simplifica el diseño de modelos de *TensorFlow* basados en redes neuronales. Una vez definida la estructura del modelo, se especifica la función de costo y su optimizador. Durante el entrenamiento se almacenan los pesos para poder escoger el de mejor rendimiento sobre los datos de validación, que representan el 20% de las muestras totales. Los modelos se comparan utilizando las métricas desarrolladas en 3.3.7, presentes en el paquete *metrics* de *Scikit Learn*⁶, valorando aquellos que obtengan mejores resultados de F-valor global.

4.3.1 Selección de arquitectura de red

Teniendo en cuenta los resultados obtenidos durante la selección de descriptores (detallados en la sección 5), los cuales son favorables al uso de espectros en escala Mel, se decide experimentar con modelos de CNN. Estos toman como entrada a un espectrograma de 128 componentes de frecuencia en escala Mel por 128 cuadros temporales, que se obtienen utilizando $N_{FFT} = 2048$, $H = 1024$ y $N_{MELS} = 128$. Para su diseño, se combinan bloques compuestos de una capa convolucional con filtros cuadrados ($f_H = f_W$) de tamaño variable, con activación Re-LU y normalización por lotes, seguida de una capa de agrupamiento de máximos. En todos los casos, se utiliza el algoritmo ADAM y una función de costo de entropía cruzada categórica, debido a su implementación exitosa en tareas de clasificación. Se diseñan principalmente tres modelos, referidos como A, B y C, cuyas arquitecturas se ilustran en la Figura 17.

⁴ Código fuente en: <https://github.com/tensorflow/tensorflow/tree/master/tensorflow>

⁵ Código fuente en: <https://github.com/keras-team/keras/tree/master>

⁶ Código fuente en: https://scikit-learn.org/stable/modules/model_evaluation

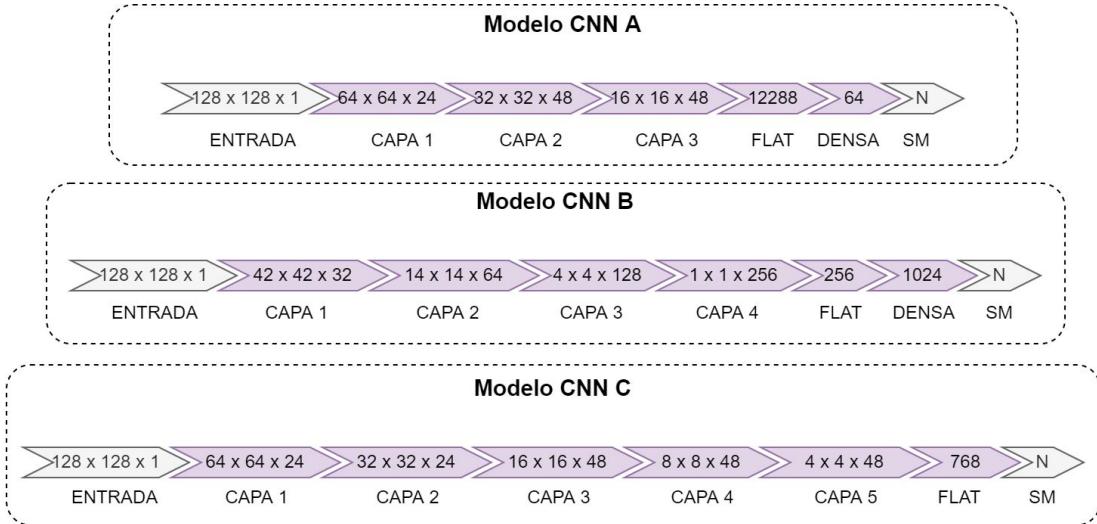


Figura 17: Esquema de diversas arquitecturas de CNN empleadas.

La arquitectura CNN-A es una versión similar al modelo propuesto por Salamon [32], quien trabaja con espectrogramas en escala Mel, utilizando bloques convolucionales (CAPA N) compuestos de normalización de lote, capa convolucional con filtros de tamaño 5×5 y agrupamiento por máximos. Tras el último bloque convolucional, se utiliza una capa de achatamiento (FLAT) que reúne los parámetros en un tensor unidimensional, el cual se acopla a una capa totalmente conectada (DENSA). Las funciones de activación son Re-LU para todas las capas menos la última capa (SM), donde se usa *softmax* para obtener un tensor de probabilidad de longitud N.

Por otro lado, la arquitectura CNN-B es una versión simplificada del modelo ganador de las competiciones DCASE-2018, diseñado por Sakashita [98], empleando la misma cantidad de filtros por capa, de tamaño 3×3 , y agrupamiento de máximos, de 3×3 . Entre las principales diferencias, su modelo se beneficia de la información binaural además de la monofónica. A partir de esta última, extrae información armónica y temporal almacenando la información en tres canales de spectrograma, en lugar de uno solo. Por último, en este trabajo se propone la arquitectura CNN-C, que utiliza menor cantidad de parámetros internos debido a la reducida cantidad de filtros. En este caso se emplean cinco bloques convolucionales con filtros de tamaño 3×3 , agrupamiento de máximos, con grupos de tamaño 2×2 y normalización de lote.

Para la selección de arquitectura de modelo se hacen pruebas utilizando TUT-COMBI junto a las bases de datos TUT-2017 y TUT-2018 procesadas.

4.3.2 Modelo final propuesto

Teniendo en cuenta los resultados obtenidos a partir de la experimentación con diversas arquitecturas de CNN, se opta por seleccionar la arquitectura CNN-C, debido a que alcanza un rendimiento similar a las otras arquitecturas CNN, pero utilizando un número reducido de parámetros (como se detalla en la Sección 5).

La estructura del modelo consiste en cinco capas convolucionales en serie, acoplada a una capa densa de salida con activación *softmax*. La resolución frecuencial y temporal del espectrograma se reduce a la mitad tras cada capa; mientras que su profundidad, es decir la cantidad de canales, aumenta. En cuanto al tamaño de los filtros, se experimenta con variantes horizontales de dimensión (1, 128), verticales de dimensión (128, 1), y cuadrados, de (2, 2) y (3, 3), los cuales extraen características tanto frecuenciales como temporales.

El tensor de salida de la red representa las probabilidades de pertenencia según la clase, a partir del cual se realiza la predicción tomando la clase de máxima probabilidad como criterio. El proceso completo para la predicción de clase de escena sonora a partir de una señal de audio se muestra en la Figura 18.

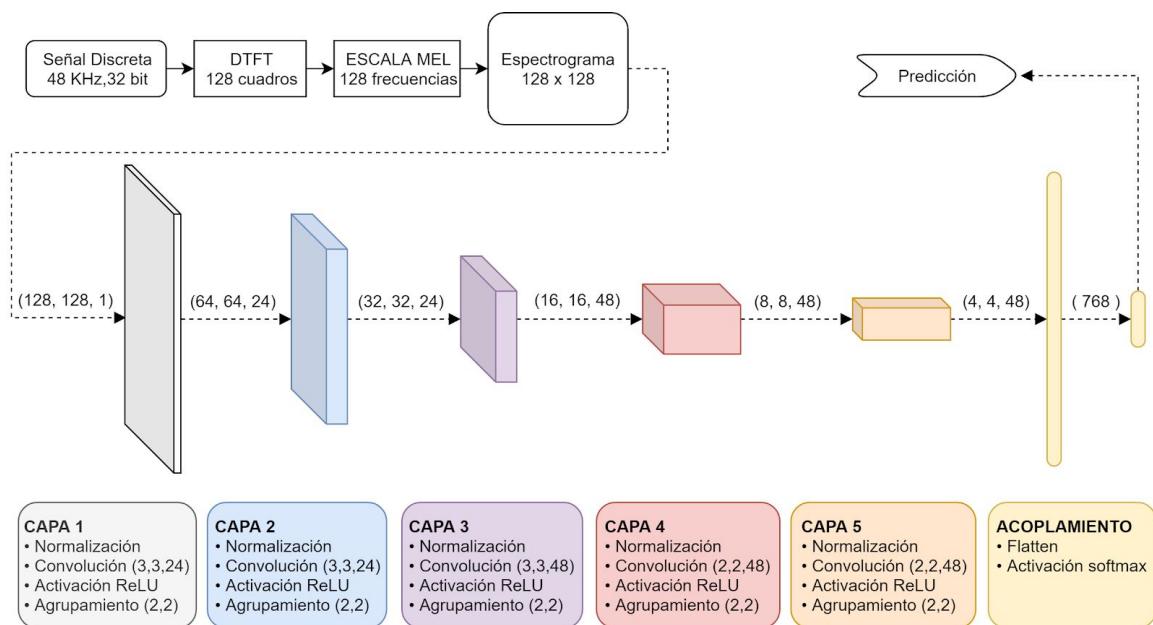


Figura 18: Esquema completo de modelo propuesto para la clasificación de escenas acústicas.

4.3.3 Técnicas de aumento de datos

A partir del modelo propuesto, se experimenta con diversas técnicas de aumento de datos, con el propósito de mejorar su rendimiento y robustez. Estas técnicas buscan generar nuevas muestras alterando las ya existentes, procurando que la etiqueta de la muestra no se vea afectada. Por ejemplo, en clasificación de imágenes, es posible generar nuevas muestras rotando las ya existentes, ya que un objeto rotado es el mismo objeto. Adicionalmente, de esta forma se propicia cierta invarianza ante la rotación.

En este trabajo se realizan pruebas con dos tipos de procesos, el primero está inspirado en el trabajo de Park et al. [99], orientado al reconocimiento del habla. La técnica consiste en el enmascaramiento de regiones aleatorias del espectrograma a través de dos franjas horizontales y verticales, de posición y ancho variable, como se muestra en la Figura 19. La cantidad total de muestras a las que se le aplica el enmascaramiento dentro de un lote es aleatoria, es decir, solo se le aplica el proceso a una porción de las muestras de lote.

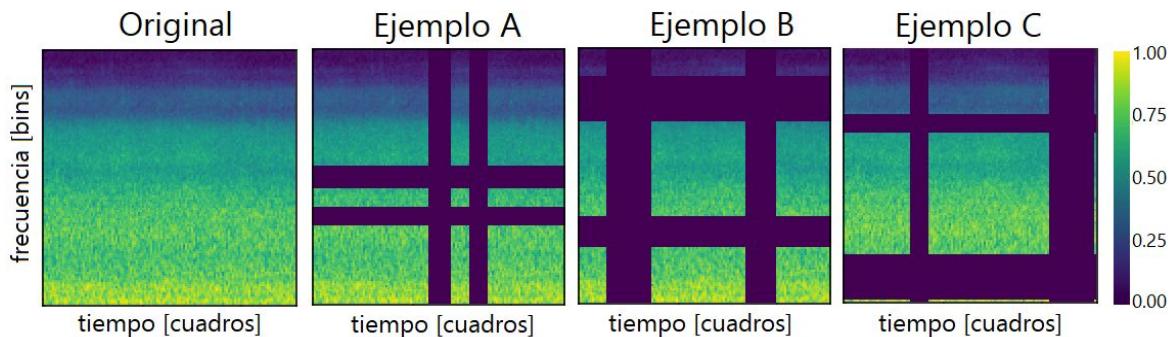


Figura 19: Ejemplos de aumento de datos utilizando enmascaramiento sobre espectrogramas.

El segundo proceso de aumentos de datos se centra en mejorar la robustez del modelo ante la variabilidad de respuesta en frecuencia de los micrófonos de dispositivos móviles. Inicialmente, para cada muestra de entrenamiento se genera un filtro pasa bajos (LPF) de topología Butterworth, limitando el orden (n) entre 1 y 20 y la frecuencia de corte normalizada (ω_c) entre 0.4 y 1, cuya función de transferencia está definida según la Ecuación 13.

$$G(\omega) = \frac{1}{\sqrt{1 + \frac{\omega}{\omega_c}^{2n}}} \quad (13)$$

A partir del filtro Butterworth se genera un HPF invirtiendo la curva de transferencia, de modo que al combinar ambos, se obtiene un BPF simétrico.

Estos filtros son lineales, de modo que para utilizarlos sobre espectrogramas en escala Mel es necesario transformarlos, lo cual se logra trasladando la curva de transferencia lineal a los centros de frecuencia del banco de filtros Mel y luego suavizando la curva a partir de filtros de Savitzky-Golay [100]. Luego, a la curva de transferencia resultante se le aplica el logaritmo para convertir a decibeles, y posteriormente se lo normaliza a un rango acotado entre 0 y 1 como se muestra en la Figura 20.

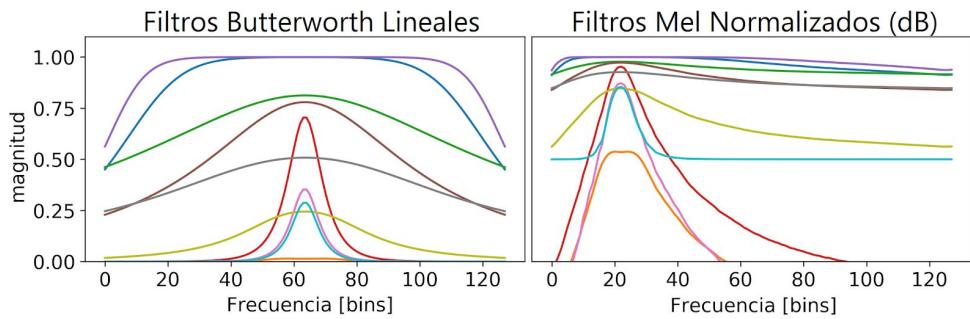


Figura 20: Generación aleatoria de filtros Butterworth en escala Mel normalizada.

Una vez transformada, la banda de paso se centra por encima de los 600 Hz. Finalmente, se extiende la curva de transferencia para cada cuadro temporal, construyendo una matriz de igual dimensión al espectrograma, de forma que el producto entre matrices resulte en el filtrado del espectrograma, como se muestra en la Figura 21.

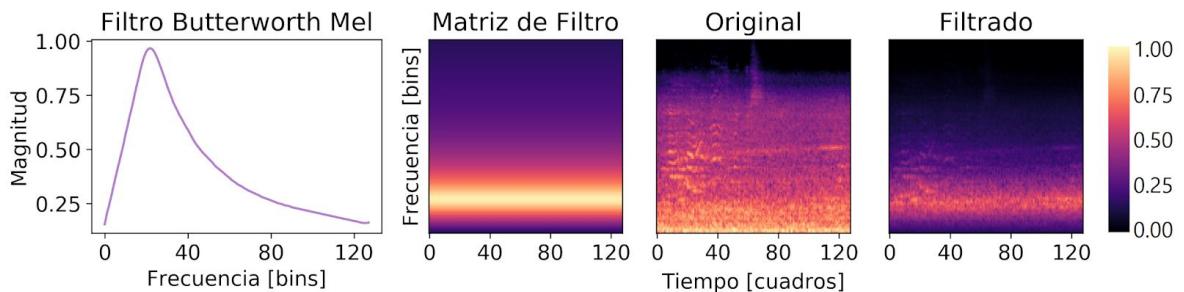


Figura 21: Ejemplo de aumento de datos por filtrado aleatorio de espectrogramas.

4.4 APLICACIÓN EN ANDROID

En este trabajo se implementa el modelo propuesto en una aplicación móvil. Se opta por el sistema operativo (OS) *Android*, debido a que es la plataforma libre más utilizada globalmente y cuenta con el soporte requerido. El desarrollo de una aplicación conlleva una serie de desafíos: en primer lugar, la adquisición y digitalización de la señal sonora capturada por el micrófono del móvil; su procesamiento digital y cálculo de espectrograma; la inferencia del modelo clasificador; y finalmente, la visualización de la predicción en una interfaz gráfica. Esto se lleva a cabo en el entorno de desarrollo integrado *Android Studio*, utilizando JAVA como lenguaje de programación.

4.4.1 Adquisición de audio

El primer desafío consiste en la adquisición de la señal de audio a través del dispositivo móvil, un proceso que en *Android* generalmente se realiza utilizando la clase estándar *MediaRecorder* o en su lugar *AudioRecord*⁷. La primera devuelve una salida de audio comprimido, no adecuada para procesamiento digital, por lo tanto, se opta por la segunda, que permite obtener el flujo continuo de muestras de entrada de audio, las cuales deben ser almacenadas temporalmente en un búfer. Se requiere una serie de parámetros relacionados con el formato de lectura, entre ellos: la fuente física de audio, la frecuencia de muestreo, la configuración de canal, el formato de codificación y finalmente, el tamaño de búfer. Este último depende de la capacidad de procesamiento del dispositivo, de modo que debe ser calculado durante la ejecución. La selección de parámetros se refleja en la Tabla 6.

Tabla 6: Configuración de parámetros de *AudioRecord* para adquisición de audio en *Android*.

Parámetro	Descripción	Configuración
audioSource	<i>Fuente de entrada de audio</i>	UNPROCESSED
sampleRateInHertz	<i>Frecuencia de muestreo [Hz]</i>	48000
channelConfig	<i>Configuración de canal</i>	CHANNEL_IN_MONO
audioFormat	<i>Formato de codificación</i>	ENCODING_PCM_FLOAT

⁷ Código fuente en: <https://developer.android.com/reference/android/media/AudioRecord>

La configuración UNPROCESSED garantiza la adquisición de muestras del micrófono evitando el preprocesamiento para voz y comunicaciones. Por otro lado, la frecuencia de muestreo, el formato de canal y la codificación se corresponden con los utilizados para extracción de descriptores durante el desarrollo del modelo predictivo.

El proceso de lectura es computacionalmente demandante, por lo que se debe utilizar un hilo de procesador independiente al de la actividad principal, el cual se encarga de los procesos relacionados a la interfaz de usuario. Esto se lleva a cabo definiendo el proceso en una clase independiente, la cual extiende la funcionalidad de *AsyncTask*⁸.

El acceso al micrófono en dispositivos *Android* requiere de la autorización del usuario, la cual es concedida al momento de la instalación de la aplicación para versiones SDK menores a 23. Sin embargo, para las versiones mayores, como es el caso de esta aplicación, el permiso RECORD_AUDIO es solicitado durante la ejecución, de modo que es necesario crear métodos locales que interactúan con el usuario para conseguir el consentimiento, el cual también se debe declarar en el archivo *manifiesto* de la aplicación.

4.4.2 Procesamiento de señales

Al finalizar la captura de audio, se tiene un vector de muestras de 32 bit, al cual se le debe calcular el espectrograma. Esto se lleva a cabo internamente en la aplicación utilizando funciones y métodos definidos en JAVA. El diagrama de procesamiento de audio se ilustra a través de la Figura 22.

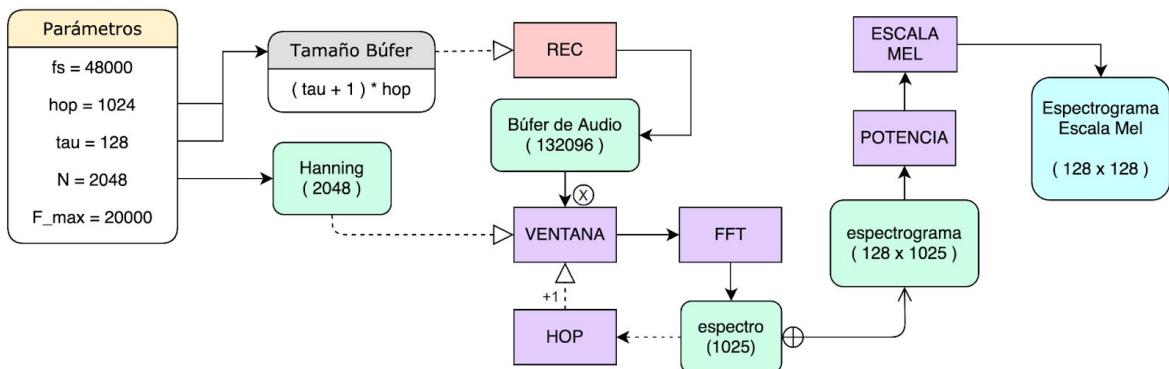


Figura 22: Esquema de procesamiento y cómputo de espectrograma en Android.

⁸ Código fuente en: <https://developer.android.com/reference/android/os/AsyncTask>

Estos procesos deben ser iguales a los realizados previamente en la etapa de desarrollo en *Python*, lo que impone un desafío, ya que, si bien *JAVA* cuenta con algunos paquetes que realizan operaciones similares, no todas las operaciones de *Python* tienen soporte. Se opta por transcribir el código fuente de las funciones que intervienen en el cálculo del espectrograma de escala Mel utilizadas en *Python* y escribirlas en *JAVA*. Para llevar esto a cabo, primero se expresan los métodos originales de *Python* en operaciones utilizando bucles más tradicionales, verificando que ambos métodos sean equivalentes; luego, se escriben en *JAVA*, respetando la sintaxis propia del lenguaje. Excepcionalmente, para el cálculo de la FFT se utiliza una versión reducida del paquete de análisis de *Minim*⁹.

Para validar la transcripción del procesamiento de la señal, se exporta un audio registrado con el dispositivo junto a su espectrograma obtenido en Android a un archivo de texto. Éste luego se lee y decodifica en *Python*, en donde paralelamente se extrae el espectrograma del audio importado, utilizando los métodos originales del paquete *librosa*. Las matrices obtenidas por ambos métodos se restan, consiguiendo una representación cuantificable de la diferencia entre ambos procesos, que se refleja en la Figura 23 con el ejemplo de una *chirp* y un sonido ambiental, comprobando que la diferencia entre métodos es despreciable (<1 %).

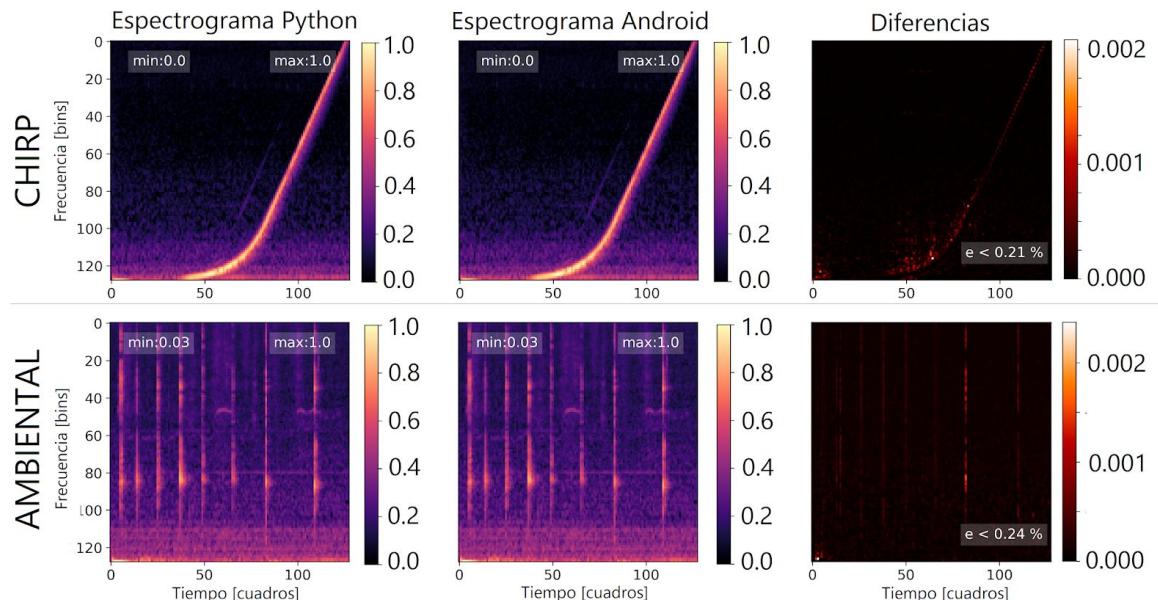


Figura 23: Validación de metodología para extracción de espectrogramas en Android.

⁹ Código fuente en: github.com/ddf/Minim/blob/master/src/main/java/ddf/minim/analysis/FFT.java

4.4.3 Inferencia

Para implementar un modelo de ANN de *TensorFlow* en una aplicación móvil, es necesario exportarlo en un formato estructurado que pueda ser interpretado y ejecutado en *Android*. Para ello, en trabajos anteriores se generaba un archivo proto-búfer (.pb) y se acudía a un objeto de *Android* llamado “*TensorFlowInferenceInterface*”. Sin embargo, en esta tesis se utiliza *TensorFlow Lite*¹⁰, una herramienta novedosa de *TensorFlow*, lanzada en 2018, que permite obtener un archivo de formato *tflite*, que contiene una versión reducida del modelo de *Keras*, con soporte de operaciones para efectuar inferencias en los sistemas operativos *Android* y *iOS*, como se muestra en la Figura 24.

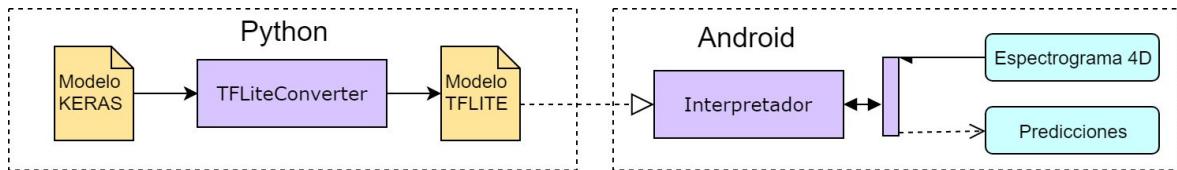


Figura 24: Esquema de implementación de red neuronal en Android.

Para utilizar el modelo en *Android*, se debe almacenar el archivo *tflite* en una carpeta dentro de la aplicación. Durante la ejecución, se utiliza el interpretador de *TensorFlow Lite* para cargar el modelo y efectuar una inferencia sobre un tensor multidimensional de entrada. Para las CNN, el modelo espera un tensor de entrada con cuatro dimensiones similar al de una imagen: (N_{MUESTRAS} , $N_{\text{HORIZONTAL}}$, N_{VERTICAL} , N_{CANALES}), donde la primera especifica la cantidad de elementos; la segunda, la dimensión horizontal; la tercera, la dimensión vertical; y la cuarta, la cantidad de canales. En este caso, utilizando una sola muestra, con 128 cuadros temporales, 128 componentes de frecuencia y 1 canal. La salida del modelo es un tensor de dimensión equivalente a la cantidad de clases, cuyas componentes reflejan la probabilidad de pertenencia a la clase, en un número de 0 a 1.

Con el fin de validar la inferencia del modelo de *TensorFlow Lite* en *Android*, se exporta una serie de espectrogramas y sus tensores de salida resultantes a un archivo de texto, que luego se importa en Python, donde se emplea el modelo original de *Keras* para realizar una inferencia sobre los mismos espectrogramas. La diferencia entre tensores en todos los casos analizados resulta menor al 1%.

¹⁰ Guía de uso en: <https://tensorflow.org/lite/guide>.

4.4.4 Estructura de interfaz de usuario

El ciclo de vida de la aplicación comienza al ejecutar la actividad principal; aquí se definen los procesos que deben ejecutarse al momento del inicio, entre ellos: el control de permisos para el acceso al micrófono y la inicialización de la interfaz de usuario.

Una vez inicializada, el usuario debe presionar un botón para que se comience a grabar el sonido ambiente. Cuando se registran más de tres segundos de audio, la aplicación internamente procesa las muestras para generar el espectrograma en escala Mel.

Tras calcular el espectrograma, se actualiza un mapa de bits en donde se lo representa gráficamente, utilizando un mapa de color de tipo gradiente de nueve puntos, predeterminado por el usuario. Inmediatamente después, se ejecutan los procesos de inferencia sobre el espectrograma utilizando la red neuronal, obteniendo como salida el tensor de probabilidades. Por defecto, se almacenan en memoria los últimos cinco tensores, de manera que, tras cada inferencia, se promedia la salida con las anteriores, estabilizando el resultado. Para realizar la predicción, se seleccionan las primeras tres clases con mayor valor, y se muestran al usuario a través de texto, ya sea en inglés o español, junto a sus respectivos porcentajes de probabilidad, reflejados también en barras horizontales.

Una vez iniciado, el ciclo se repite indefinidamente, hasta que el usuario detenga el ciclo o bien detenga la aplicación. La interacción entre los objetos de la interfaz gráfica de la aplicación se muestra en la Figura 25, a continuación. Nota: ver ANEXO IV para detalles sobre el código desarrollado para la aplicación de Android.

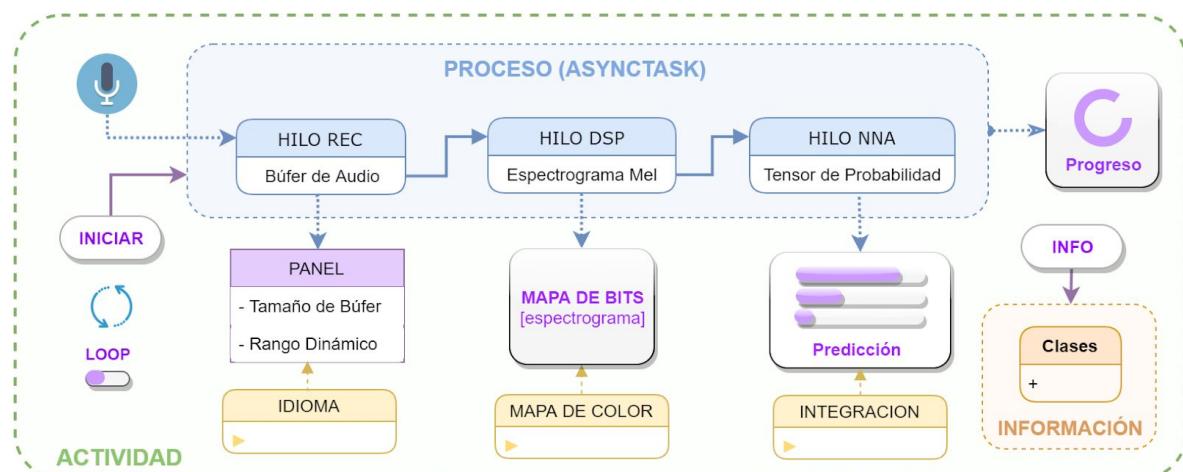


Figura 25: Esquema de interacción entre objetos de la interfaz de usuario y procesos de fondo.

5. RESULTADOS

5.1 PRUEBAS CON DESCRIPTORES

En la Figura 26 se comparan las medias de los descriptores con información en componentes de frecuencia para las clases de las bases TUT-2017 y TUT-2018; y en la Figura 27 según la media de los descriptores globales.

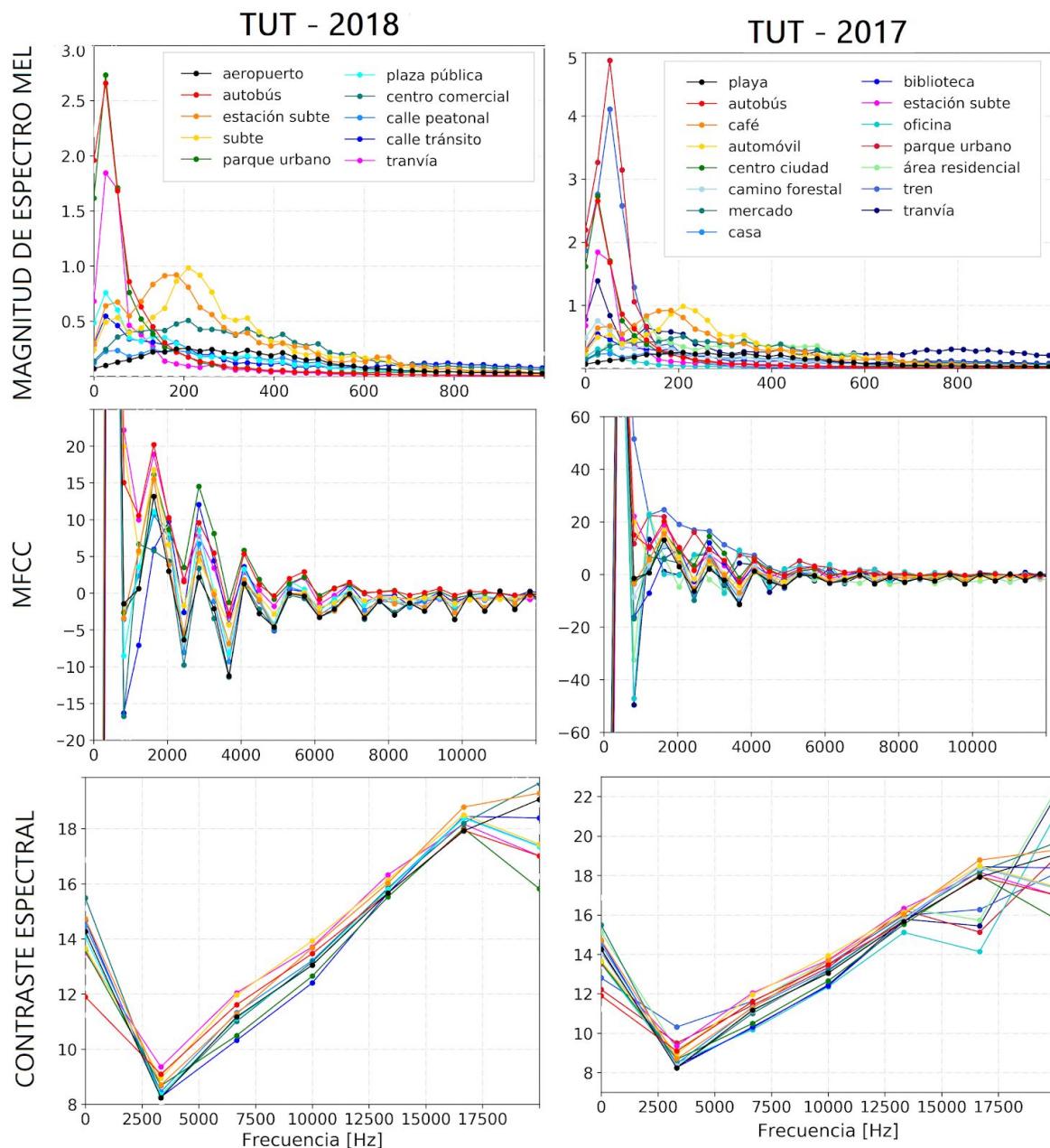


Figura 26: Medias de espectro Mel, MFCC y SCO por frecuencia para TUT-18 (a) y TUT-17 (b).

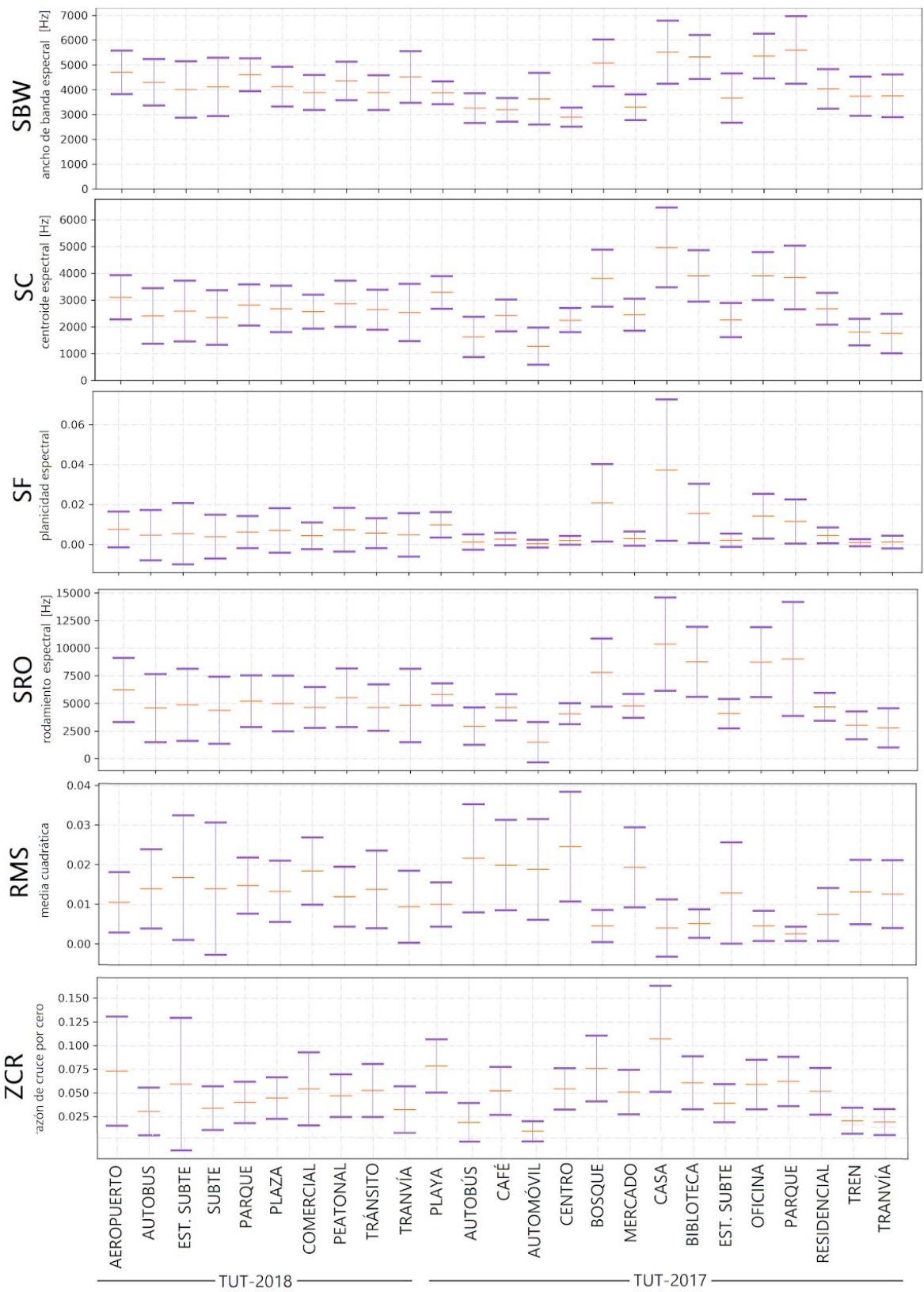


Figura 27: Medias y desvíos estándar para descriptores globales de las clases de las bases TUT.

En la Tabla 7 se reflejan los resultados de la media sobre el F-valor de todas las clases (F-valor Global) y el F-valor mínimo, correspondiente a la clase con peor desempeño, obtenidos utilizando diversos descriptores con la red DNN-0, como se describe en la Sección 4.2.2. En la tabla, la combinación de SF, ZCR, SBW, SRO y SC se indica como “Combinado”. En la Figura 28 se muestran los resultados de F-valor para cada clase.

Tabla 7: F-valor global y mínimo para descriptores de bases TUT utilizando DNN-0.

Descriptores	TUT-2017		TUT-2018	
	F-Valor Global	F-Valor Mín	F-Valor Global	F-Valor Mín
SF	0.13	0.00	0.15	0.00
ZCR	0.15	0.00	0.22	0.00
SBW	0.17	0.00	0.18	0.00
SRO	0.21	0.00	0.21	0.00
SC	0.22	0.00	0.19	0.00
Combinado	0.46	0.27	0.38	0.22
SCO	0.61	0.38	0.52	0.35
MFCC	0.76	0.57	0.56	0.32
Espectro Mel	0.89	0.75	0.69	0.52

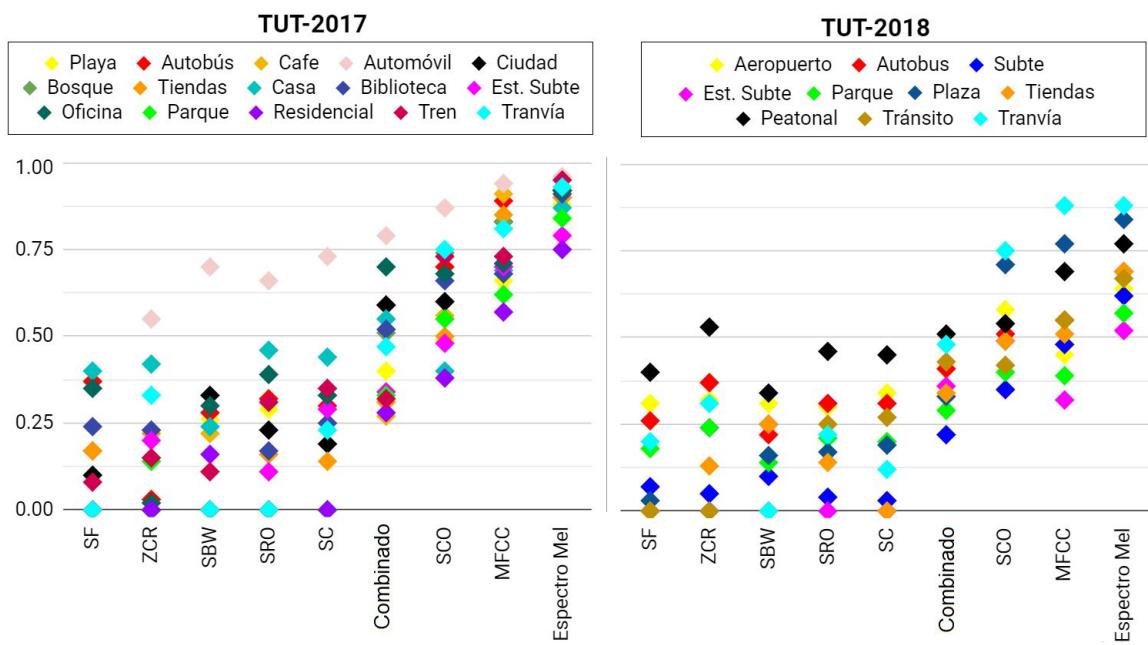


Figura 28: Comparación de F-valor por clase para DNN-0 utilizando descriptores de bases TUT.

Por otro lado, en la Tabla 8 se muestran los resultados de F-valor (F1) global asociados a las variaciones de la arquitectura DNN, aunque en este caso sólo para los descriptores de MFCC y espectro Mel. En la Figura 29 se grafican los resultados de F-valor por clase.

Tabla 8: F-valor global para diversas arquitecturas DNN utilizando MFCC y espectro Mel.

Variación de Arquitectura	Capas Ocultas	Neuronas Por Capa	TUT-2017		TUT-2018	
			MFCC (F1)	MELS (F1)	MFCC (F1)	MELS (F1)
DNN A	2	25	0.65	0.85	0.57	0.65
DNN B	2	100	0.80	0.89	0.60	0.71
DNN C	2	200	0.83	0.92	0.63	0.75
DNN D	1	50	0.63	0.78	0.54	0.65
DNN E	3	50	0.79	0.88	0.59	0.69
DNN F	4	50	0.79	0.89	0.61	0.72

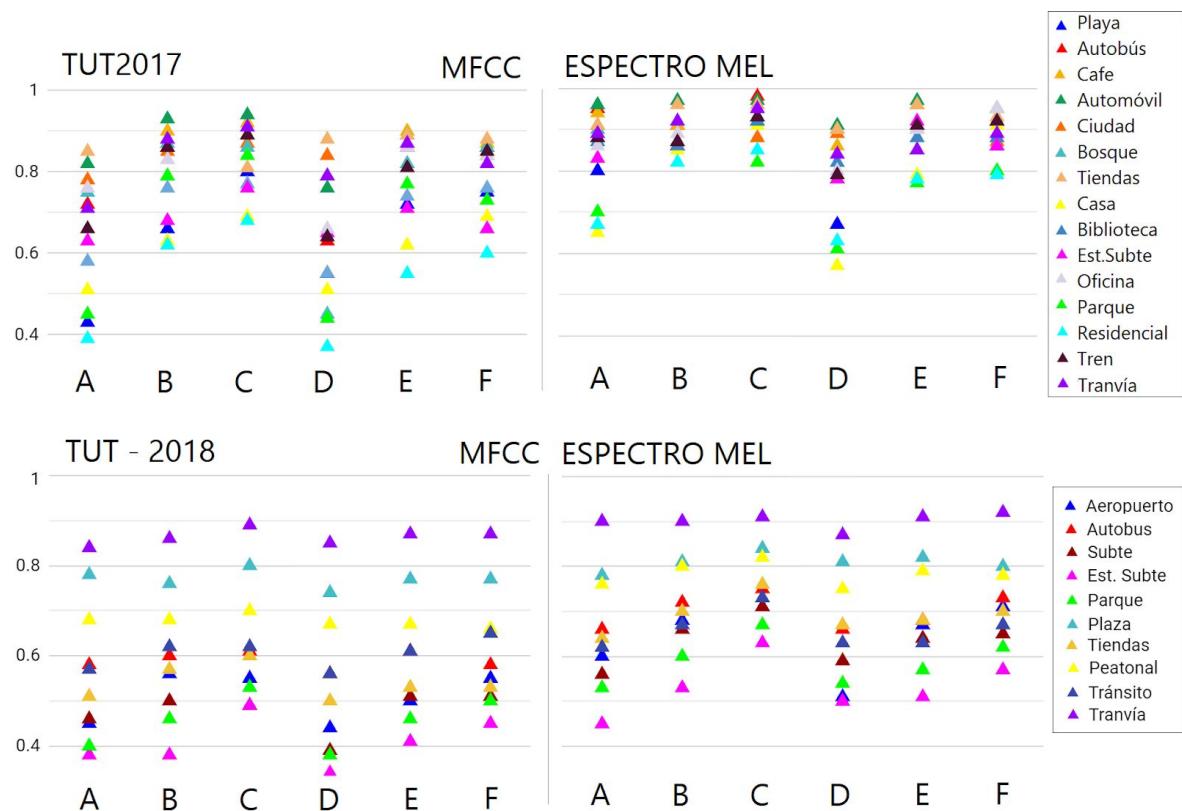


Figura 29: F-valor por clase para diversas arquitecturas DNN utilizando MFCC y espectro Mel.

5.2 PRUEBAS CON MODELOS

5.2.1 Pruebas con redes convolucionales

En la Tabla 9 se muestran los resultados de F-valor para las arquitecturas de redes convolucionales planteadas en la Sección 4.3.1, utilizando por un lado las bases de datos TUT-2017 y TUT-2018 con el fin de compararlos con trabajos previos, y por el otro, utilizando TUT-COMBI, propuesta en la Sección 4.1.2. Adicionalmente, en la Figura 30 se detallan los resultados por clase.

Tabla 9: F- valor global y mínimo para diversas arquitecturas CNN utilizando base de datos TUT.

Base de Datos	Arquitectura	Nº Parámetros	F-Valor Global	F-Valor Mínimo
TUT-2017	CNN-A	819395	0.986	0.97
	CNN-B	668307	0.981	0.95
	CNN-C	46699	0.981	0.96
TUT-2018	CNN-A	819070	0.783	0.68
	CNN-B	663182	0.824	0.75
	CNN-C	42854	0.830	0.75
TUT-COMBI	CNN-A	819135	0.898	0.80
	CNN-B	664207	0.886	0.79
	CNN-C	43623	0.895	0.79

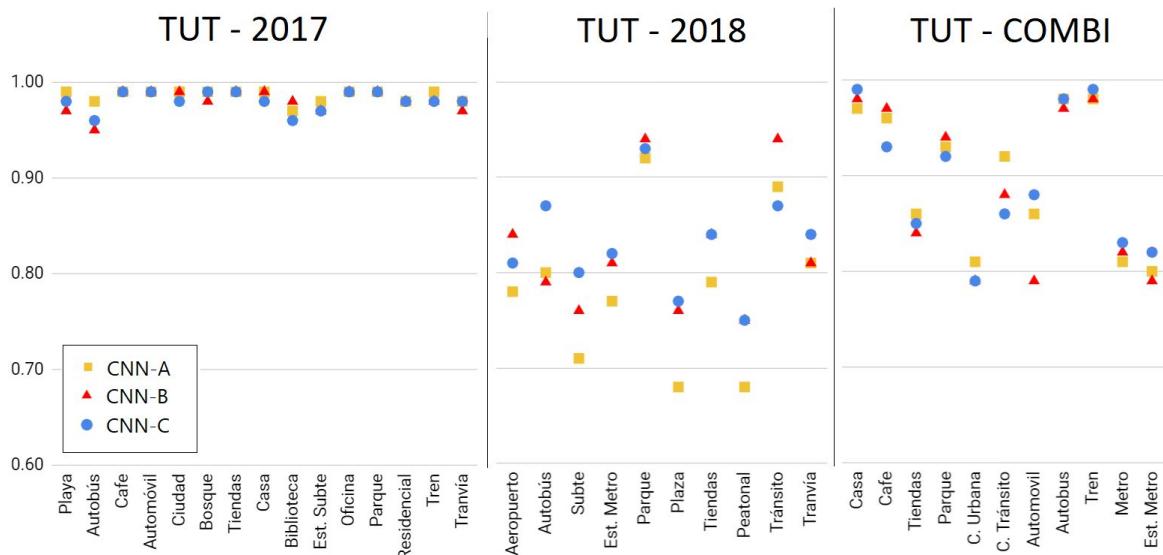


Figura 30: F-valor por clase para diversas arquitecturas de redes convolucionales.

Se experimenta variando la forma de los filtros de la arquitectura CNN-C, intentando favorecer la extracción de características frecuenciales y temporales. Las variaciones consisten en utilizar filtros horizontales (1, N) y verticales (N, 1), donde N es equivalente a la dimensión del spectrograma en cada capa. Los resultados de F-valor globales y específicos por clase se muestran en la Tabla 10 y en la Figura 31, respectivamente.

Tabla 10: F-valor global de modelos utilizando filtros convolucionales verticales y horizontales.

Modelo	Tipo de Filtros	F-Valor Global	F-Valor Mínimo
Rectangular	cuadrados	0.89	0.79
Temporal	verticales	0.73	0.56
Frecuencial	horizontales	0.81	0.70

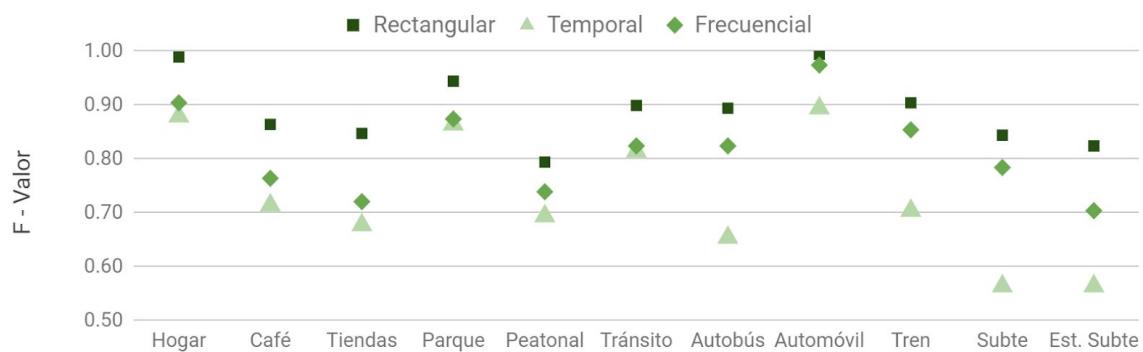


Figura 31: F- valor por clase para variantes de modelos CNN con filtros verticales y horizontales.

Las bases de datos LITIS-ROUEN, TUT-2017 Y TUT-2018 tienen en común la clase *estación de metro y bus*, de manera que cada modelo entrenado con una de éstas puede ser evaluado a partir de las dos restantes, en lo que se denomina “inter-validación”. Esto permite tener una noción de la capacidad de generalización de los modelos según la bases de datos utilizada para su entrenamiento. Los resultados de aciertos porcentuales utilizando CNN-C se muestran en la Tabla 11.

Tabla 11: Resultados de aciertos (%) utilizando CNN-C para inter-validación de bases de datos

MODELO ENTRENADO	DATOS DE EVALUACIÓN		
	LITIS-ROUEN	TUT-2017	TUT-2018
CNN LITIS-ROUEN	90.3%	5.8%	5.4%
CNN TUT-2017	50.0%	99.9%	45.7%
CNN TUT-2018	60.1%	61.8%	91.1%

5.2.2 Pruebas con aumento de datos

En estas pruebas se evalúan las técnicas de aumento de datos por enmascaramiento y filtrado aleatorio de spectrogramas. Las pruebas se realizan sobre la arquitectura CNN-C, utilizando particiones de las bases de datos TUT-2017, TUT-2018 y TUT-COMBI. Los resultados de F-valor y AUC mínimo para cada caso se muestran en la Tabla 12. Adicionalmente, en la Figura 32 se muestran los resultados de F-valor por clase.

Tabla 12: F-valor y AUC para modelo CNN-C utilizando técnicas de aumento de datos.

Base de Datos	Aumento de Datos	F-Valor Global	F-Valor Mín	AUC Mín
TUT-2017	Sin DA	0.981	0.96	0.94
	Enmascaramiento	0.985	0.97	0.98
	Filtros Aleatorios	0.983	0.97	0.98
TUT-2018	Sin DA	0.830	0.75	0.86
	Enmascaramiento	0.877	0.80	0.94
	Filtros Aleatorios	0.841	0.73	0.96
TUT-COMBI	Sin DA	0.895	0.79	0.95
	Enmascaramiento	0.936	0.86	0.96
	Filtros Aleatorios	0.931	0.86	0.97

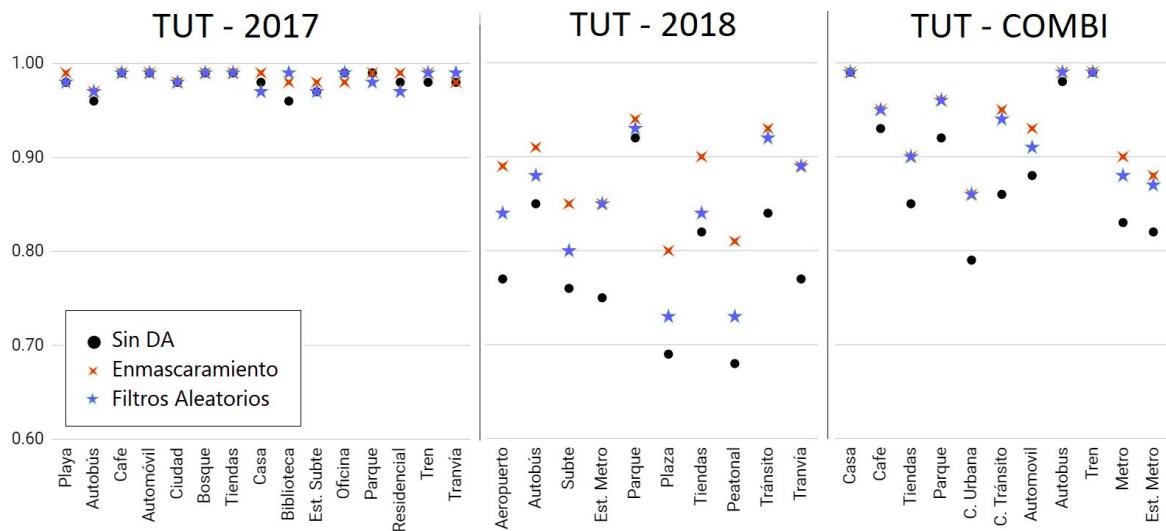


Figura 32: F-valor por clase para arquitectura CNN-C utilizando técnicas de aumento de datos.

5.3 EVALUACIÓN FINAL

A partir de las grabaciones de escenas acústicas registradas en Buenos Aires, se extraen los spectrogramas en escala Mel y se genera una base de datos de evaluación (BA-DB). Las muestras obtenidas se etiquetan siguiendo la estructura de clases utilizada en TUT-COMBI. Esta nueva base de datos se utiliza para evaluar el modelo final, con arquitectura CNN-C y aplicación de técnicas de aumento de datos como enmascaramiento y filtros aleatorios. Similarmente, se realizan pruebas recortando los spectrogramas, eliminando las componentes en frecuencia inferiores a 100 Hz y superiores a 10 kHz, con el fin de evaluar su influencia en el rendimiento del modelo. Las métricas globales de precisión, sensibilidad y F-valor para cada variante del modelo se muestran en la Tabla 13, mientras que en la Figura 33 se muestra una comparación del F-valor por clase.

Tabla 13: Métricas globales de evaluación para variantes de modelo final utilizando BA-DB.

Proceso	Precisión	Sensibilidad	F-Valor Global	F-Valor Min
Ninguno	0.58	0.49	0.44	0.10
Enmascaramiento	0.54	0.45	0.42	0.03
Filtros Aleatorios	0.61	0.57	0.54	0.14
Recorte	0.57	0.32	0.29	0.05
Todos los anteriores	0.52	0.44	0.39	0.17

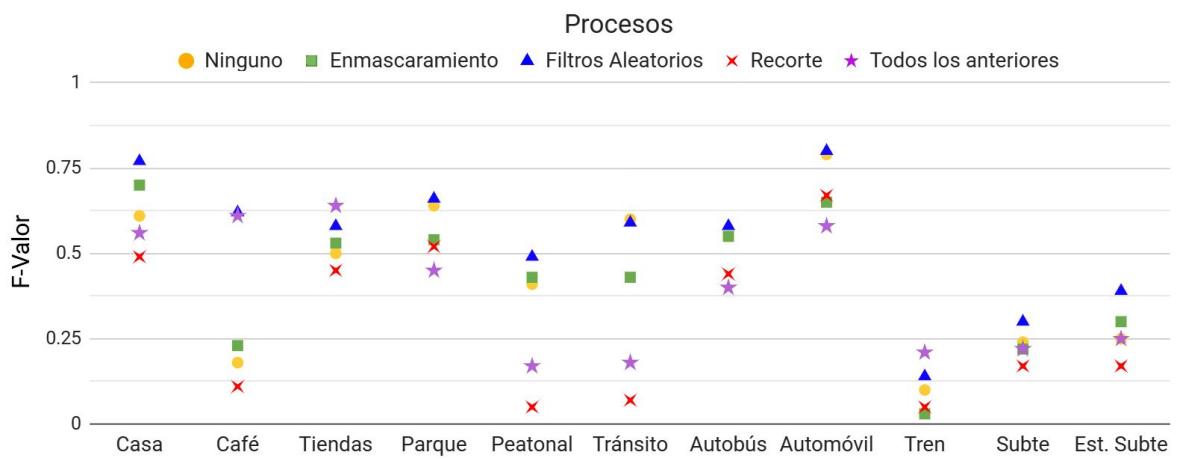


Figura 33: F-valor por clase para evaluación de variantes del modelo final utilizando BA-DB.

En la Figura 34 se comparan las matrices de confusión (x: estimadas, y: verdaderas) obtenidas a partir de la evaluación de las variantes del modelo final, utilizando por un lado la partición reservada para evaluación de TUT-COMBI y, por otro lado, el conjunto BA-DB. Además, en la Figura 35 se muestra la curva ROC y AUC para el modelo con variante de filtros aleatorios.

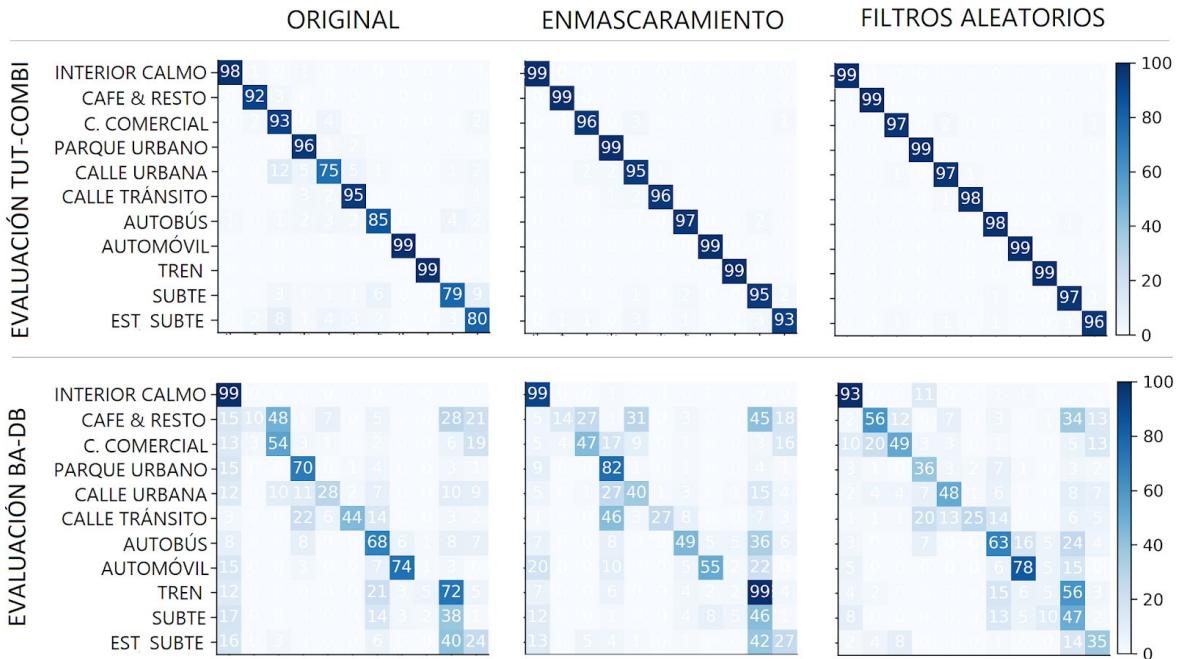


Figura 34: Matrices de confusión para evaluación de variantes de aumento de datos.

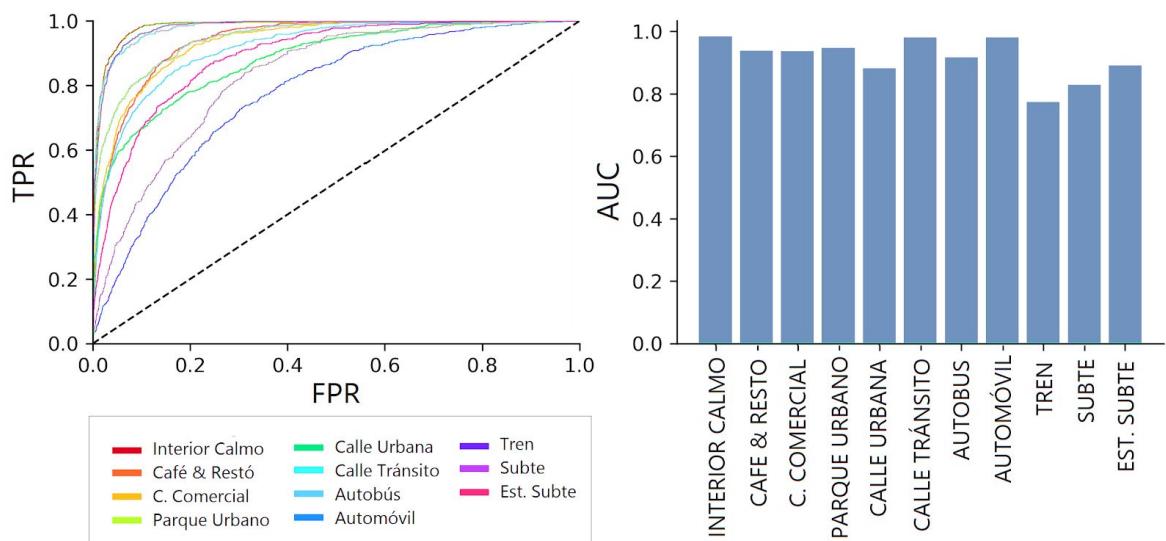


Figura 35: Curva ROC y AUC para modelo final utilizando técnica de filtros aleatorios.

6. DISCUSIÓN

En esta sección se realiza un análisis de los resultados obtenidos en las pruebas, así como de la evaluación final del modelo implementado en la aplicación móvil.

6.1 COMENTARIOS SOBRE LAS BASES DE DATOS

El modelo predictivo desarrollado se genera a partir del entrenamiento supervisado con muestras provenientes de las bases de datos de acceso abierto de mayor tamaño existente a la fecha, específicamente elaboradas para la investigación académica. No obstante, trabajando con ellas, se hallaron ruidos de interferencia y golpes en una porción significativa de las muestras de ciertas clases, llevando el aprendizaje de patrones equivocados que resultaron en problemas durante el entrenamiento de los modelos iniciales. Esto obligó a realizar la inspección manual exhaustiva de las señales y el descarte de las muestras que presentaban ruidos o eventos sonoros no correspondientes a la escena en cuestión.

Debido a que la distribución de ruidos no es homogénea entre clases, se eliminaron muestras de forma despareja, comprometiendo el balance entre clases. Es menester mencionar que la limpieza en conjunto con la aplicación de HPF, mejoró significativamente el rendimiento del modelo y su capacidad de generalización al evaluarlo con BA-DB.

Debido a que las muestras consisten en grabaciones de varios minutos de duración, en un número limitado de locaciones, presentan poca diversidad, perjudicando el entrenamiento de modelos y su capacidad de generalización. Por otro lado, las clases definidas no contemplan el universo completo de escenas acústicas, sino una selección limitada, de forma que fue necesario recurrir a la combinación de varios conjuntos de datos. La combinación de muestras de bases de datos generadas con diferentes metodologías no es ideal, ya que se corre el riesgo que los modelos se ajusten en función de los patrones que diferencian las bases de datos, en lugar de los que caracterizan a la clase. Por este motivo, a pesar de que LITIS-ROUEN presenta la mayor variedad de clases, no se tuvo en cuenta para realizar combinaciones, siendo que sus muestras presentan una envolvente espectral muy distinta al de las bases TUT (ver Figura I.2 del ANEXO I).

6.2 DESCRIPTORES

6.2.1 Análisis de valores medios

A partir de los resultados presentados en la Figura 26 para las medias de descriptores con información en componentes de frecuencias, extraídos a partir de TUT-2017 y TUT-2018, se puede ver que las componentes más determinantes en la separación de clases se encuentran en bajas frecuencias para los MFCC y el espectro Mel, ya que para frecuencias altas todas las clases presentan un comportamiento similar. Se destaca la similitud de los espectros Mel de las clases relacionadas al transporte y *parque urbano* en ambas bases de datos, lo que se atribuye a la presencia de un ruido de fondo de baja frecuencia en los *parques urbanos*. Asimismo, se destacan los máximos locales centrados en 200 Hz existentes en los espectros Mel de las clases *metro*, *estación de metro*, *automóvil*, *café* y *centro comercial*, cuyo denominador común es la presencia de voces humanas interactuando. Se observa que los MFCC presentan un patrón de oscilación similar entre clases, aunque sutilmente distinto, mientras que las medias de SCO se diferencian especialmente en las bandas de alta frecuencias.

Al analizar la Figura 27, se puede apreciar que ninguno de los descriptores globales por sí solo es suficiente para llevar a cabo una separación total entre clases basándose exclusivamente en su media y su desvío. Se observa que las clases de TUT-2017 generalmente asociadas con escenas acústicas tranquilas, como *bosque*, *casa*, *biblioteca*, *oficina* y *parque*, presentan un ancho de banda espectral, SC, SRO y SF muy superior al resto. Por otro lado, como era de esperarse, el RMS es un descriptor válido para separar escenas acústicas tranquilas de las escenas más “ruidosas”. Si bien se puede concluir que estos descriptores son útiles para diferenciar las clases asociadas a escenas tranquilas de las demás, se advierte que no son necesariamente determinantes para diferenciar dichas escenas entre sí. En cuanto a las clases relacionadas con el transporte, como son *autobús*, *automóvil*, *tren* y *tranvía*, se observa que la media y desvíos se asemejan en la mayoría de sus descriptores, presentando valores de SC, SRO y ZCR inferiores al resto de las clases.

6.2.2 Análisis de métricas

A partir de la Tabla 7, se comprueba que ninguno de los descriptores globales (SF, ZCR, SBW, SRO y SC) es suficiente por sí solo para clasificar correctamente a todas las clases de las bases TUT-2017 y TUT-2018. Este hecho es evidenciado por un F-valor global pobre, cercano a 0.2, y un F-valor mínimo equivalente a cero. Sin embargo, al combinar estos descriptores, la precisión y sensibilidad del modelo aumentan significativamente, alcanzando un F-valor global cercano a 0.4 y un F-valor mínimo superior a 0.22.

Por otro lado, los descriptores con información por componentes de frecuencia, como es el caso del SCO, MFCC y el espectro en escala Mel, presentan una mejora de F-valor por sobre los descriptores globales, como se muestra en la Figura 28. Se encuentra que los mejores resultados se obtienen al utilizar el espectro de escala Mel, seguidos por los MFCC, tanto para TUT-2017 como para TUT-2018. Si bien los MFCC han sido señalados en trabajos previos como los más idóneos para la clasificación de escenas acústicas utilizando arquitecturas DNN, por encima del espectro Mel, en las pruebas llevadas a cabo no sucede así. Esto se refleja también en la Tabla 8, donde el espectro Mel continúa siendo superior a los MFCC, independientemente de la arquitectura de red empleada, escalando su rendimiento conforme se incrementa la cantidad de parámetros.

6.3 MODELOS

6.3.1 Análisis de arquitecturas CNN

Los modelos que utilizan arquitecturas CNN inspiradas en Salamon (CNN-A) o Sakashita (CNN-B), poseen más de 500.000 parámetros, en cambio, el modelo propuesto en esta tesis (CNN-C), cuenta con menos de 45.000. Al evitar el uso de capas densas, el número de parámetros es incluso menor que el de la DNN de mayor tamaño. Los resultados en la Tabla 9 muestran que las tres arquitecturas presentan un F-valor global similar, a pesar de la diferencia en la cantidad de parámetros, siendo la arquitectura CNN-C la más eficiente, aunque se advierte que existen clases en donde los otros modelos son superiores. Con la arquitectura CNN-C se obtiene un F-valor global superior a 0.89 y un F valor mínimo de 0.79, lo que representa una mejora sobre los modelos DNN y una mejora en eficiencia de parámetros frente a los otros modelos CNN.

6.3.2 Análisis de filtros convolucionales

Se escoge la arquitectura CNN-C para llevar a cabo la experimentación con tamaños de filtros, cuyos resultados, reflejados en la Tabla 10, muestran que los modelos que implementan filtros cuadrados (en este caso 3 x 3) son preferibles a los filtros exclusivamente verticales o exclusivamente horizontales. Esto se puede atribuir a que los primeros extraen patrones frecuenciales y temporales en simultáneo, mientras los otros se enfocan en una sola dimensión. Se destaca que se obtuvieron mejores resultados utilizando filtros frecuenciales por sobre los filtros temporales, lo que implica que la red le da más relevancia al análisis estático del espectro que al análisis de su dinámica temporal.

6.3.3 Análisis de inter-validación de datos

Entre otras pruebas llevadas a cabo utilizando la arquitectura CNN-C, se encuentra la inter-validación de bases de datos. La Tabla 11 refleja que los modelos entrenados con las bases TUT y evaluados en las otras dos obtienen un porcentaje de exactitud superior al 45% en las clases evaluadas, mientras que, en los modelos entrenados exclusivamente a partir de LITIS-ROUEN se obtienen predicciones de muy baja exactitud (menor al 6%). Por ende, se evidencia que los datos de TUT-2017 y TUT-2018 son más adecuados para obtener modelos con mayor capacidad de generalización a otras bases de datos.

6.3.2 Análisis de pruebas con aumento de datos

Los resultados reflejados en la Tabla 12 sugieren que tanto la técnica de enmascaramiento como la de filtros aleatorios imprimen una mejora marginal sobre el modelo base, con una sutil ventaja para los modelos entrenados utilizando la técnica de enmascaramiento. Sin embargo, se observa que los valores de AUC obtenidos favorecen a los modelos entrenados mediante técnicas de filtrado aleatorio. Se repara en que estos resultados se obtienen utilizando una partición reservada de las bases de datos de entrenamiento, de modo que estas pruebas no son concluyentes para cuantificar una mejora en la capacidad de generalización de los modelos. Para ello se requiere el uso de una nueva base de datos de evaluación, como la generada en esta tesis, empleada en las pruebas finales.

6.3.4 Análisis de pruebas finales

Se comparan las variantes del modelo CNN-C que implementan técnicas de aumento de datos y otras técnicas de entrenamiento, utilizando la base de datos de evaluación BA-DB generada a partir de escenas acústicas registradas en CABA. Los resultados de estas pruebas brindan información acerca de la capacidad de generalización que alcanza el modelo en función de las técnicas implementadas, evaluando su efectividad.

Al evaluar inicialmente el modelo CNN-C utilizando BA-DB se obtiene un F-valor global de 0.44, significativamente inferior al 0.89 obtenido previamente al usar la partición de datos de TUT-COMBI como conjunto de evaluación. Por un lado, esto sugiere que las muestras de escenas acústicas presentes en las bases TUT, registradas en ciudades europeas, no serían totalmente representativas de las escenas acústicas de Buenos Aires. Por otro lado, es previsible que las características propias del transductor del teléfono móvil con el que se registra BA-DB difieren de las de los dispositivos utilizados durante la creación de las bases TUT. Adicionalmente, el resultado obtenido se puede atribuir a problemas intrínsecos de las bases TUT, relacionados con la metodología empleada para su registro, detallados en la sección 6.1.

Contrariamente a los resultados discutidos en 6.3.2, la Tabla 13 muestra que, evaluando con BA-DB, la técnica de enmascaramiento no supone una mejora por sobre el modelo base, sino que presenta una desmejora de los resultados en todas las métricas (precisión, sensibilidad, F-valor) y para todas las clases. Sin embargo, la técnica de filtros aleatorios aplicada por sí sola es la que presenta los mejores resultados, alcanzando una precisión mayor al 0.6 y un F-valor global de 0.54.

Por otro lado, se analiza la influencia del recorte del espectrograma para frecuencias inferiores a 100 Hz y superiores a 10 kHz, empleado con la finalidad de forzar al modelo a reconocer patrones en un rango adecuado a la respuesta en frecuencia típica de los dispositivos móviles. En este caso, los resultados muestran una desmejora sobre el modelo base, sugiriendo que las componentes de frecuencia recortadas contienen información valiosa para el modelo, y deben ser preservadas. La combinación de todas estas técnicas en simultáneo tampoco logra una mejora, debido a que el enmascaramiento y el recorte impactan de forma negativa en la capacidad de reconocimiento de patrones.

El modelo entrenado con filtros aleatorios es el de mejores resultados y por este motivo se lo adopta como “modelo final”. Al analizar su matriz de confusión (Figura 34), se observa una confusión moderada entre *calle-peatonal* y *calle-tránsito*, las cuales presentan espectrogramas menos estacionarios y con mayor presencia de eventos sonoros impredecibles. Las clases con peor rendimiento son *tren* y *metro*, ya que se confunden mutuamente, especialmente cuando se trata de trenes eléctricos. El modelo rara vez predice *tren*, lo cual podría deberse a que los trenes registrados en Buenos Aires no son similares a los europeos, los cuales presentan sonidos de platos y cubiertos. Debido a que el modelo final está basado en espectrogramas de escala Mel, su principal desventaja es que lo vuelve muy sensible a la presencia de voces, las cuales están excesivamente representadas en las escenas de metro de las bases TUT. Esto repercute negativamente en el desempeño del modelo, ya que tiende a predecir *metro* al captar voces de campo cercano.

6.4 COMENTARIOS SOBRE LA EVALUACIÓN DE LA APLICACIÓN

Por cuestiones de practicidad, el desempeño predictivo de la aplicación móvil se evalúa de forma indirecta: por un lado, se validan los procesos internos de *Android* para la extracción de espectrogramas e inferencia con *TensorFlow-Lite*, como se detalla en las secciones 4.4.2 y 4.4.3, obteniendo diferencias menores al 1% respecto a las funciones originales de *librosa* y *TensorFlow*. Por otro lado, se genera la base de datos BA-DB a partir del registro de escenas acústicas de Buenos Aires, utilizando un único dispositivo móvil (Moto-G4), con el fin de evaluar el modelo final desarrollado en *TensorFlow*. De manera que los resultados obtenidos a partir de esta evaluación son un equivalente representativo del desempeño de la aplicación al usar este dispositivo en Buenos Aires, sin tener que repetir las pruebas para evaluar variaciones del modelo.

En el caso de los resultados obtenidos utilizando BA-DB para la evaluación del modelo final, se obtiene un F-valor global de 0.55, con un desvío de 0.20. La exactitud varía según la escena, obteniendo mejor desempeño en la clasificación de *automóvil* e *interior-caldo* ($F_1 > 0.75$), aunque con bajo rendimiento para la clasificación de *tren* y *metro* ($F_1 < 0.35$). Para extraer estos resultados a otros dispositivos móviles sería necesario extender BA-DB añadiendo muestras registradas con esos dispositivos.

7. CONCLUSIONES

A partir de esta investigación, se desarrolló un modelo para el reconocimiento automático de escenas acústicas utilizando una red neuronal convolucional. El modelo fue implementado en una aplicación para dispositivos móviles que permite identificar diversos entornos acústicos, que incluyen espacios interiores, espacios públicos tales como plazas, calles peatonales o con tránsito, y medios de transporte, entre los que se encuentran los automóviles, autobuses, el tren y el metro. El código de la aplicación desarrollada es de acceso libre y puede instalarse en la mayoría de los dispositivos del mundo, ya que está desarrollado en Android.

El desarrollo del sistema fue posible gracias a la existencia de bases de datos de escenas acústicas de gran tamaño, disponibles de forma abierta, y a los recientes avances en el aprendizaje automático, en particular, para su implementación en aplicaciones móviles. No obstante, se remarca que fue necesario realizar un pre-procesamiento exhaustivo de los datos para mejorar el desempeño de los modelos.

Los resultados de las métricas obtenidas a partir del entrenamiento con TUT indican que la arquitectura CNN-C desarrollada en este trabajo compite con otros modelos más complejos, como el recreado a partir del modelo diseñado por Sakashita, el cual se considera el estado del arte para la clasificación de escenas acústicas utilizando TUT-2018. Por otro lado, con el fin de extender la variedad de clases, en este trabajo se generó un conjunto de datos combinados, que fue utilizado para el entrenamiento del modelo final, basado en la arquitectura CNN-C.

Como producto de esta investigación, se generó además una base de datos de evaluación de escenas acústicas registradas en la Ciudad Autónoma de Buenos Aires con un único dispositivo móvil, que fue utilizada para validar el modelo desarrollado en *TensorFlow* en situaciones más realistas y para evaluar distintas técnicas de aumento de datos, entre ellas de filtrado aleatorio, con la que se obtuvo una mejora de hasta 10 puntos porcentuales de F-valor, superando el 55% global. Las clases con mejor desempeño en las pruebas con este dispositivo fueron casa y automóvil ($F_1 > 0.75$), y las de peor rendimiento fueron *tren* y *metro* ($F_1 < 0.30$).

8. TRABAJOS FUTUROS

En esta sección se presentan posibles mejoras que podrían contribuir al desarrollo de nuevos modelos de reconocimiento automático de escenas acústicas y su uso en dispositivos móviles.

En primer lugar, las bases de datos más importantes existentes a la fecha presentan una variedad acotada de clases, enfocadas en los entornos sonoros existentes en ciudades europeas y grabadas con una pequeña selección de dispositivos móviles. Se propone el desarrollo de gran cantidad de datos de entrenamiento, obtenidos con mayor variedad de dispositivos móviles, que contemplen las características inherentes a las escenas acústicas de ciudades latinoamericanas, especialmente para los medios de transporte. Por ejemplo, sería útil extender BA-DB, incorporando nuevas locaciones registradas con otros dispositivos móviles. Alternativamente, se propone experimentar con escenas acústicas generadas sintéticamente.

En segundo lugar, se sugiere realizar pruebas con rangos diferentes de tamaños de FFT y tamaños de salto para la generación de espectrogramas, con el fin de estudiar en mayor profundidad la influencia de la resolución frecuencial y temporal en el desempeño de modelos para el reconocimiento de patrones sonoros ambientales. Adicionalmente, se propone incorporar en múltiples canales del espectrograma información complementaria a la magnitud que puede ser extraída a partir del audio de escenas acústicas.

En tercer lugar, se podrían explorar nuevas técnicas de aumento de datos para mejorar la generalización de los modelos. Una posibilidad es perfeccionar la caracterización de la respuesta en frecuencia de los dispositivos móviles con el fin de generar nuevos datos a través de filtros que simulan los efectos del transductor.

Por último, se propone impulsar el desarrollo de aplicaciones en Android que se beneficien del reconocimiento de escenas acústicas para la configuración automática del dispositivo y sus funciones como, por ejemplo, el control automático del volumen de timbre según el entorno acústico en el que se encuentra el usuario.

BIBLIOGRAFÍA

- [1] Yuo, K., Hwang, T., Wang, H., Combination of autocorrelation-based features and projection measure technique for speaker identification, IEEE Trans, Speech Audio Process, 13, 565–574, (2005).
- [2] Wang, Y., Zhao, Z., A Noise-robust Speech Recognition System Based on Wavelet Neural Network. In Proceedings of the Third International Conference on AICI, Volume Part III, Taiyuan, China, 392–397, (2011).
- [3] Benetos, E., Kotti, M., Kotropoulos, C., Musical Instrument Classification using Non-Negative Matrix Factorization Algorithms and Subset Feature Selection. In Proceedings of the 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Toulouse, France, 5, V:221–V:225, (2006).
- [4] Tzanetakis, G., Cook, P. R., Musical genre classification of audio signals. IEEE Trans. Speech Audio Process, 10, 293–302, (2002).
- [5] Cheng, H.T., Yang, Y., Lin, Y., Liao, I., Chen, H.H., Automatic chord recognition for music classification and retrieval. In Proceedings of the 2008 IEEE ICME, Hannover, 1505–1508, (2008).
- [6] Bello, J. P., Mydlarz, C., Salamon, J., Sound analysis in smart cities, Computational Analysis of Sound Scenes and Events. Springer International Publishing, p 373-397, (2017).
- [7] Hornstein, J., Lopes, M., Victor, J. S., Lacerda, F., Sound Localization for Humanoid Robots - Building Audio-Motor Maps based on the HRTF. IEEE/RSJ International Conference on Intelligent Robots and Systems. (2006).
- [8] Ellis, D., Virtanen, T., D.Plumbley, M., Computational Analysis of Sound Scenes and Events. Springer International Publishing AG., (2017).
- [9] Alías, F., Socorro, J., Sevillano, X., A Review of Physical and Perceptual Feature Extraction Techniques for Speech, Music and Environmental Sounds. GTM - Grup de recerca en Tecnologies Mèdia, La Salle-Universitat Ramon Llull, 30, Barcelona, Spain, (2016).

- [10] Barchiesi, D., Giannoulis, D., Stowell, D., and Plumbley, M. D., Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Processing Magazine*, (2015).
- [11] Stowell, D., Giannoulis, D., Benetos, E., Lagrange, M., Plumbley, M., Detection and classification of acoustic scenes and events, 10, 1733–1746, (2015).
- [12] Beritelli, F., Grasso, R. A pattern recognition system for environmental sound classification based on MFCCs and neural networks, In Proceedings of the ICSPCS, 15-17, (2008).
- [13] Muhammad, G., Alghathbar, K., Environment Recognition from Audio Using MPEG-7 Features, Proceedings of the 4th International Conference on Embedded and Multimedia Computing, Jeju, Korea, 1–6, (2009).
- [14] Tsau, E., Kim, S.H., Kuo, C.C.J., Environmental sound recognition with CELP-based features, Proceedings of the 10th International Symposium on Signals, Circuits and Systems, Iasi, Romania, 3, 1–4, (2011).
- [15] Valero, X., Alías, F., Classification of audio scenes using Narrow-Band Autocorrelation features, Proceedings of the 20th EUSIPCO, 2012–2019, (2012).
- [16] Heittola, T., Mesaros, A., Eronen, A., Virtanen, T., Audio context recognition using audio event histograms, 18th European Signal Processing Conference, 1272–1276, (2010).
- [17] Rakotomamonjy, A., Gasso, G., Histogram of gradients of time-frequency representations for audio scene classification, *IEEE/ACM Trans. Audio Speech Lang.*, 142–153, (2015).
- [18] Chu, S., Narayanan, S.S., Kuo, C.J., Environmental Sound Recognition with Time-Frequency Audio Features, *IEEE Trans. Audio Speech Lang.*, 17, 1142–1158, (2009).
- [19] De Coensel, B., Botteldooren, D., A model of saliency-based auditory attention to environmental sound, 20th International Congress on Acoustics (ICA), 1–8, (2010).
- [20] Cauchi, B., Non-negative matrix factorization applied to auditory scene classification. Master's Thesis, ATIAM UPMC/IRCAM, (2011).
- [21] Geiger, J., Schuller, B., Rigoll, G., Large-scale audio feature extraction and SVM for acoustic scene classification, Proceedings of the 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), 1–4, (2013).

- [22] Benetos, E., Lagrange, M., Dixon, S., Characterisation of acoustic scenes using a temporally constrained shift-invariant model, Proceedings of Conference on Digital Audio Effects, (2012).
- [23] Kobayashi, T., Ye, J., Acoustic feature extraction by statistics based local binary pattern for environmental sound classification, Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 3052–3056. IEEE, (2014).
- [24] Battaglino, D., Lepauloux, L., Pilati, L., Evansi, N. Acoustic context recognition using local binary pattern codebooks, Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 1–5, (2015).
- [25] Sigtia, S., Stark, A.M., Krstulovic, S., Plumbley, M.D., Automatic environmental sound recognition: performance versus computational cost. *Audio Speech Lang. Process*, (2016).
- [26] Petetin, Y., Laroche, C., Mayoue, A., Deep neural networks for audio scene recognition. 23rd EUSIPCO, 125–129, (2015).
- [27] Piczak, K.J., Environmental sound classification with convolutional neural networks, In IEEE International Workshop on Machine Learning for Signal Processing, (2015).
- [28] Tokozume Y. T. H., Learning environmental sounds with end-to-end convolutional neural network. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2712–2725, (2017).
- [29] Bae, S.H., Choi, I., Kim, N.S., Acoustic scene classification using parallel combination of LSTM and CNN, Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016), 11–15, (2016).
- [30] Mesaros, A., Heittola, T., and Virtanen, T., TUT database for acoustic scene classification and sound event detection, EUSIPCO, (2016).
- [31] Dai, W. ASR with Deep Learning. Carnegie Mellon University, (2016).
- [32] Salamon, J., Bello, J.P., Deep convolutional neural networks and data augmentation for environmental sound classifications, *IEEE Signal Process*, 3, 279–283, (2017).
- [33] Gygi, B., Factors in the Identification of Environmental Sounds. Ph.D. Thesis, Indiana University, (2001).
- [34] Ando, Y., A theory of primary sensations and spatial sensations measuring environmental noise. *J. Sound Vib*, 3–18, (2001).
- [35] Peltonen, V., Tuomi, J., Klapuri, A., Huopaniemi, J., Sorsa, T. Computational Auditory Scene Recognition, Proceedings of the 2002 IEEE ICASSP, 2, 1941-1944, (2002).

- [36] Cai, R., Lu, L., Hanjalic, A., Zhang, H., Cai, L., A flexible framework for key audio effects detection and auditory context inference, *Audio Speech Lang.*, 14, 1026–1039, (2006).
- [37] Lee, K., Hyung, Z., Nam, J., Acoustic scene classification using sparse feature, learning and event-based pooling, 2013 IEEE, Workshop on Applications of Signal Processing to Audio and Acoustics, 1–4, (2013).
- [38] Imoto, K., Ohishi, Y., Uematsu, H., Ohmuro, H.: Acoustic scene analysis based on latent acoustic topic and event allocation, 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), 1–6. IEEE, (2013).
- [39] Dennis, J., Tran, H.D., Chng, E.S., Image feature representation of the subband power distribution for robust sound event classification. *Audio Speech Lang*, 367–377, (2013).
- [40] Battaglino, D., Lepauloux, L., Pilati, L., Evansi, N., Acoustic context recognition using local binary pattern codebooks, Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 1–5, (2015).
- [41] Bisot, V., Essid, S., Richard, G., HOG and subband power distribution image features for acoustic scene classification, Proceedings of European SPC, 719–723, (2015).
- [42] Rakotomamonjy, A., Gasso, G., Histogram of gradients of time-frequency, representations for audio scene classification, *IEEE/ACM Trans. Audio Speech Lang. Process.* 142–153, (2015).
- [43] Lostanlen, V., Andén, J., Binaural scene classification with wavelet scattering. Technical Report, DCASE2016 Challenge, (2016).
- [44] Bisot, V., Serizel, R., Essid, S., and Richard, G. Acoustic scene classification with matrix factorization for unsupervised feature learning, *Acoustics, Speech and Signal Processing (ICASSP)*, 2016 IEEE International, (2016).
- [45] Valenti, M., Diment, A., Parascandolo, G., Squartini, S., Virtanen, T., DCASE 2016 acoustic scene classification using convolutional neural networks, Proceedings of the DCASE2016 Workshop, 95–99, (2016).
- [46] Xu, Y., Huang, Q., Wang, W., Jackson, P.J.B., Plumbley, M.D., Fully DNN-based multi-label regression for audio tagging, Proceedings of the DCASE2016 Workshop, pp. 105–109, (2016).
- [47] Defagot, G, A., Generación de vocabularios y extracción de tópicos para la detección de escenas acústicas mediante T-SNE y LDA. Universidad de Buenos Aires, Facultad de Ciencias exactas y Naturales, Dpto de Computación, (2018).

- [48]** Mattia, R., Feese, S., Amft, O., Braune, N., Martis, S., Troster, G., AmbientSense: A realtime ambient sound recognition system for smartphones, Pervasive Computing and Communications Workshops (PERCOM Workshops), International Conference, (2013).
- [49]** Pillos, A., Alghamidi, K., Alzamel, N. O., Pavlov, V., Machanavajhala, S., A realtime environmental sound recognition system for the Android OS, (2016).
- [50]** Karol J., Piczak, ESC: Dataset for Environmental Sound Classification, <https://doi.org/10.7910/DVN/YDEPUT>, Harvard Dataverse, (2015).
- [51]** Pires, IM., Garcia N. M., Pombo, N., Flórez-Revuelta, F., From Data Acquisition to Data Fusion: A Comprehensive Review and a Roadmap for the Identification of Activities of Daily Living Using Mobile Devices, NCBI, PubMed, (2016).
- [52]** Salamon, J., Jacoby, C., Bello, J.P., A dataset and taxonomy for urban sound research, Proceedings of the ACM International Conference on Multimedia, 1041–1044, (2014).
- [53]** Piczak, K.J., Dataset for environmental sound classification, Proceedings of the International Conference on Multimedia (ACM), 1015–1018, (2015).
- [54]** Foster, P., Sigtia, S., Krstulovic, S., Barker, J., Plumbley, M.D., CHIME-home: a dataset for sound source recognition in a domestic environment, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), (2015).
- [55]** Mesaros,A., Heittola, T., Virtanen, T., TUT database for acoustic scene classification and sound event detection, 24th European Signal Processing Conference 2016, 1128–1132, (2016).
- [56]** Mesaros, A., Heittola, T., Virtanen, T., A multi-device dataset for urban acoustic scene classification, Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE-2018), 9–13, (2018).
- [57]** Rakotomamonjy, A., Gasso, G., Histogram of gradients of Time-Frequency representations for audio scene detection, Technical report, HAL, (2014).
- [58]** Nyquis, H., Certain Topics in Telegraph Transmission Theory, Proceedings of the IEEE. (1928).
- [59]** Waggner, B., Waggner, W. N., Waggner, W. M., Pulse Code Modulation Techniques, Springer Science & Business Media. (1995).
- [60]** Rao. K.R., Kim D. N, Hwang. J., Fast Fourier Transform - Algorithms and Applications. Springer Science & Business Media, (2011).

- [61] Cooley, J., Lewis, P., Welch, P., The finite Fourier transform. IEEE Transactions on Audio and Electroacoustics, p 77 - 85, (1969).
- [62] Heinzel, G., Rudiger, A., Schilling, R., Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new flat-top windows. Max-Plank-Institut fur Gravitationsphysik, Hannover, (2002).
- [63] Mitrovic, D., Zeppelzauer, M., Breiteneder, C., Features for content-based audio retrieval. *Adv. Comput.*, 78, 71–150, (2010).
- [64] Chen. C. H., Signal processing handbook, Dekker, New York, p531 (1988).
- [65] Klapuri, A., & Davy, M. (Eds). Signal processing methods for music transcription, Springer Science & Business Media, 5, (2007).
- [66] Peeters, G., A large set of audio features for sound description (similarity and classification) in the CUIDADO project, CUIDADO I.S.T. Project Report, (2004).
- [67] Dubnov, Shlomo, Generalization of spectral flatness measure for non-gaussian linear processes, *IEEE Signal Processing Letters*, 11, (2004).
- [68] Peeters, G., A large set of audio features for sound description in the CUIDADO PROJECT, I.S.T, Report, (2004).
- [69] Jiang, Ning, D., Lu, L., Jiang, H., Zhang, Hua-Tao, J., Cai, L. H., Music type classification by spectral contrast feature, *Multimedia and Expo, ICME'02*, vol. 1, pp.113-116, (2002).
- [70] Smith, S. S., Volkmann, J., Volkmann, N., Edwin B., A scale for the measurement of the psychological magnitude pitch, *Journal of the Acoustical Society of America*, 8, 185–190, (1937).
- [71] Slaney, M., Auditory Toolbox: A MATLAB Toolbox for Auditory Modeling Work, Technical Report, version 2, Interval Research Corporation, (1998).
- [72] Mermelstein, P., Distance measures for speech recognition, psychological and instrumental, *Pattern Recognition and Artificial Intelligence*, 374–388, (1976).
- [73] Slaney, M., Auditory Toolbox: A MATLAB Toolbox for Auditory Modeling Work. Technical Report, version 2, Interval Research Corporation, (1998).
- [74] Bishop, C. M., Pattern Recognition and Machine Learning, Springer, (2006).
- [75] Wang, J., Perez, L., The Effectiveness of Data Augmentation in Image Classification using Deep Learning. Stanford University, (2017).
- [76] Russell, S., Norvig, P., Artificial Intelligence: A Modern Approach. Malaysia, Pearson Education Limited. 709, (2016).

- [77] Keogh E., Mueen A, Curse of Dimensionality. In: Sammut C., Webb G.I., Encyclopedia of Machine Learning and Data Mining. Springer, Boston, MA, (2017).
- [78] Pourrajabi, M., Moulavi, D., Campello, R. J. G. B., Zimek, A., Sander, J., Goebel, R., Model Selection for Semi-Supervised Clustering, Proceedings of the 17th International Conference on Extending Database Technology (EDBT), 331–342, (2014).
- [79] Byvatov, E., Fechner U., Sadowski, J., Schneider, G., Comparison of Support Vector Machine and Artificial Neural Network Systems for Drug Classification, Journal of Chemical Information and Modeling, (2003).
- [80] Altman, N. S., An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 3, 175–185, (1992).
- [81] Nielsen, M. A, Neural Networks and Deep Learning, Determination Press, (2015).
- [82] Mimlitz. Z., Using Artificial Neural Networks to Analyze Trends of a Global Aircraft Fleet, Viasat Intern Projects, (2017).
- [83] Rosenblatt, F., The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, 6, 386-408, (1958).
- [84] Krizhevsky, A., Sutskever, I., Hinton, G. E., Imagenet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems* 25, 1097-1105, (2012).
- [85] Simonyan, K., Zisserman, A., Very deep convolutional networks for large-scale image recognition, preprint arXiv, 1409-1556, (2014).
- [86] Han, Y., Park, J., Lee, K., Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification, Workshop on Detection and Classification of Acoustic Scenes and Events, (2017).
- [87] Punyawiwat, P., Wattanapenpaiboon, N., Interns Explain CNN, Data Wow Blog - Intelligent Outsourcing Platform and Machine Learning, (2018).
- [88] Pons, J., Lidy, T., Serra, X., Experimenting with Musically Motivated Convolutional Neural Networks, In 14th International Workshop on Content-Based Multimedia Indexing (CBMI), (2016).
- [89] Mathworks, Deep Learning, Redes Neuronales Convolucionales.
la.mathworks.com/solutions/deep-learning/convolutional-neural-network.html.

- [90] Boer, P. T., Kroese, D. P., Mannor, S., Rubinstein, R. Y., A Tutorial on the Cross-Entropy Method, Faculty of Electrical Engineering, Mathematics & Computer ScienceDesign and Analysis of Communication Systems, (2005).
- [91] Rumelhart, D.E., Hinton, G. E., Williams, R. J., Learning internal representations by error propagation, Parallel distributed processing, MIT Press, 1, 318-362, (1986).
- [92] Ioffe, S., Szegedy, C., Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Google Inc, (2015).
- [93] Caruana, R., Lawrence, S., Giles, L., Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping, Advances in neural information processing systems, 402-408, (2000).
- [94] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning, (2014).
- [95] Powers. D. M. W., Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation, Journal of Machine Learning Technologies. 1, 37–63, (2011).
- [96] Krüger, F., Activity, Context, and Plan Recognition with Computational Causal Behaviour Models. University of Rostock. (2016).
- [97] Fawcett, T., An introduction to ROC analysis, Pattern Recognition Letters, 27, 861-874. (2006).
- [98] Sakashita, Y., Aono, M., Acoustic Scene Classification by Ensemble of Spectrograms based on adaptive temporal divisions, Detection and Classification of Acoustic Scenes and Events, (2018).
- [99] Park, D. S., Chan, W., Zhang, Y., Chiu, C., Zoph, B., SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. Cornell University, (2019).
- [100] Savitzky, A., Golay, M. J., E., Smoothing and Differentiation of Data by Simplified Least Squares Procedures, ACS Publications, (1964).

ANEXOS

ANEXO I: BASES DE DATOS Y DESCRIPTORES DE ESCENAS SONORAS

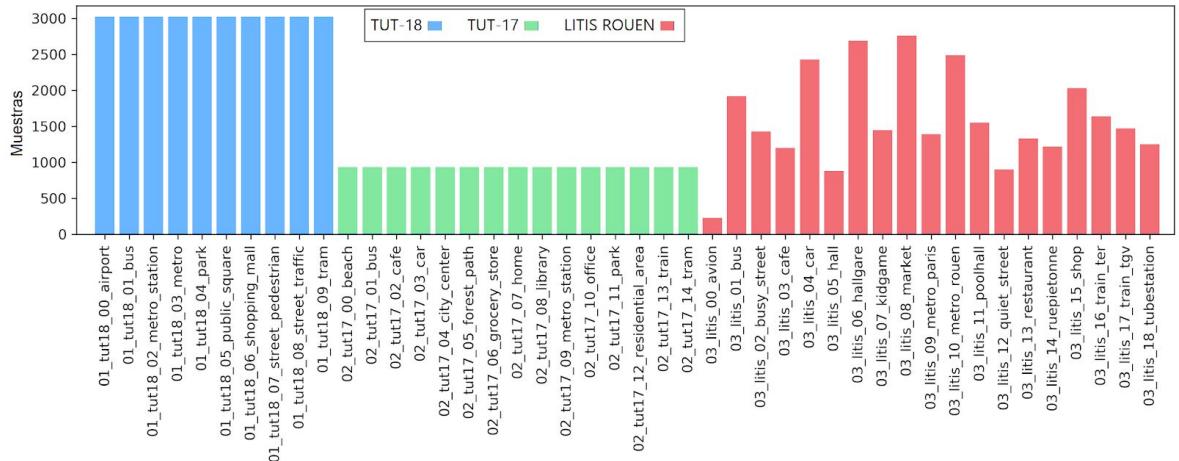


Figura I.1: Balance de bases de datos más importantes para clasificación de escenas acústicas.

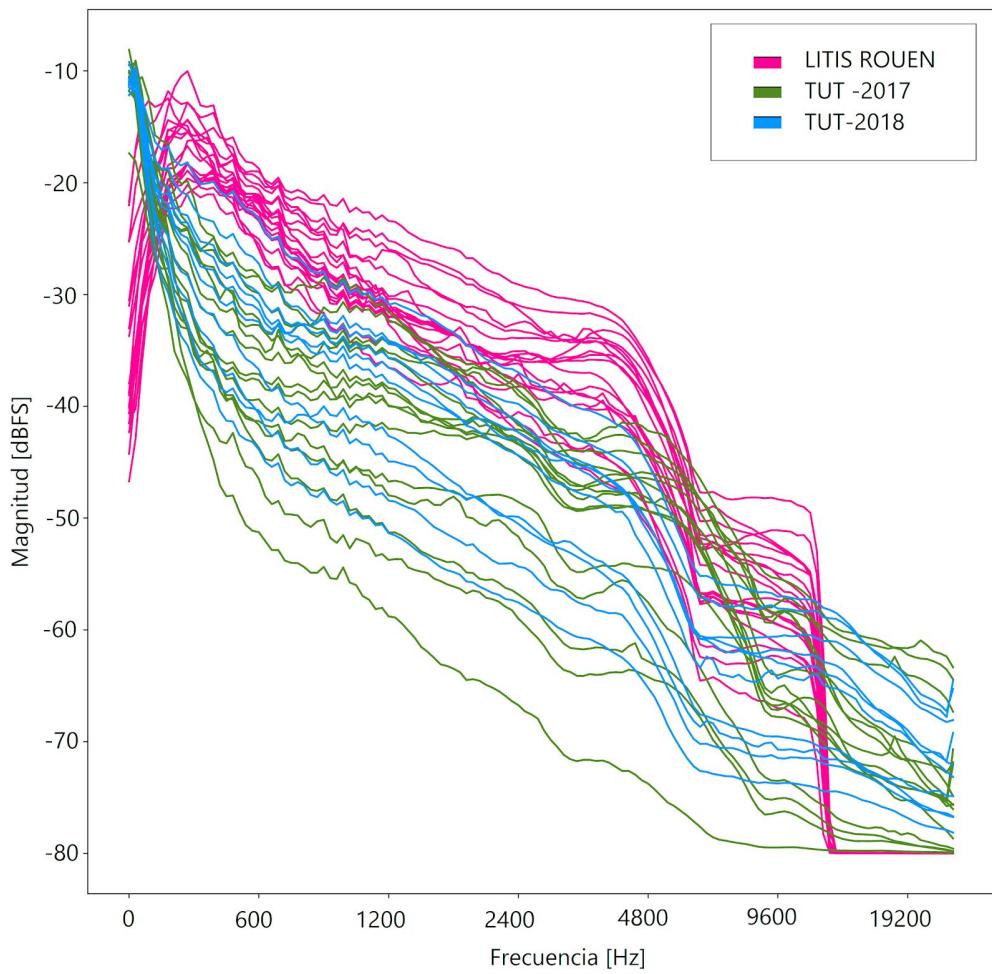


Figura I.2: Comparación de medias de envolvente espectral para bases de datos estudiadas.

Tabla I.1: Clasificación de descriptores para voz (V), música (M) y ambiente (A)

DESCRIPTORES	Taxonomía		T. Previos		
	Tipo	Categoría	V	M	A
4 Hz modulation energy	Perceptual	Frecuencial	■	■	
Amplitude descriptor	Físico	Temporal		■	
Auditory filter bank temporal envelopes	Perceptual	Frecuencial		■	
Auditory image model	Perceptual	Imagen			
Auditory saliency map	Perceptual	Otro		■	
Autocorrelation function features	Perceptual	Temporal	■	■	
Band periodicity	Físico	Temporal		■	
Bandwidth	Físico	Frecuencial	■	■	■
Beat histogram	Físico	Temporal		■	
Beat spectrum/spectrogram	Físico	Temporal		■	
Beat tracker	Físico	Temporal		■	
Chroma energy distribution normalized statistics	Físico	Frecuencial		■	
Chroma-gram	Físico	Frecuencial		■	
CM AM sensitivity in the inferior culliculus	Perceptual	Frecuencial			
Code excited linear prediction features	Físico	Frecuencial	■		■
Complex cepstrum	Físico	Cepstral	■	■	
Computational models for auditory receptive fields	Perceptual	Espectral			
Cyclic beat spectrum	Físico	Temporal		■	
Eigenspace	Físico	Otro			■
Entropy	Físico	Frecuencial	■		
Fundamental frequency	Físico	Frecuencial	■	■	■
Gabor functions	Perceptual	Wavelet	■		
GammaChirp filter banks	Perceptual	Cepstral	■		
Gammatone cepstral coefficients	Perceptual	Cepstral	■		■
Gammatone wavelet features	Perceptual	Wavelet		■	
Generalized Perceptual Linear Prediction	Perceptual	Cepstral	■		
Greenwood Function cepstral coefficients	Perceptual	Cepstral	■		■
Group delay function	Físico	Frecuencial	■	■	
Harmonic-to-Noise Ratio	Físico	Frecuencial	■	■	
Harmonicity	Físico	Frecuencial	■	■	■
Hurst parameter features	Físico	Wavelet	■		
Inharmonicity	Físico	Frecuencial	■	■	
Integral loudness	Perceptual	Frecuencial		■	
Jitter	Físico	Frecuencial	■	■	
Joint acoustic and modulation frequency features	Perceptual	Frecuencial		■	
Line spectral frequencies	Físico	Frecuencial	■	■	
Linear Prediction Cepstrum Coefficients	Físico	Cepstral	■	■	■
Linear prediction coefficients	Físico	Frecuencial	■		
Linear prediction zero-crossing ratio	Físico	Temporal	■		
Long-term analysis of short term timbre features	Perceptual	Frecuencial		■	
Loudness	Perceptual	Frecuencial		■	
Mel Frequency Cepstral Coefficients	Perceptual	Cepstral	■	■	■
Mel Frequency DiscreteWavelet Coefficients	Perceptual	Wavelet	■		
Modified group delay function	Físico	Frecuencial	■		
Modulation spectrogram	Perceptual	Frecuencial	■	■	
MP-based Gabor features	Físico	Wavelet	■	■	■
MPEG-7 audio waveform	Físico	Temporal		■	
MPEG-7 log attack time	Físico	Temporal		■	■
MPEG-7 spectral timbral descriptors	Físico	Frecuencial		■	
MPEG-7 spectrum envelope and normalized	Físico	Frecuencial	■	■	■

MPEG-7 temporal centroid	Físico	Temporal		<input checked="" type="checkbox"/>
Multiscale spectro-temporal modulations	Perceptual	Espectral	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Narrow-band autocorrelation function features	Perceptual	Temporal		<input checked="" type="checkbox"/>
Noise-robust audio features	Perceptual	Cepstral		
Octave-based Spectral Contrast	Físico	Frecuencial		<input checked="" type="checkbox"/>
Perceptual Linear Prediction	Perceptual	Cepstral	<input checked="" type="checkbox"/>	
Perceptual Tonality	Perceptual	Frecuencial		
Perceptual wavelet packets	Perceptual	Wavelet	<input checked="" type="checkbox"/>	
Pitch histogram	Físico	Frecuencial		<input checked="" type="checkbox"/>
Pitch profile	Físico	Frecuencial		<input checked="" type="checkbox"/>
Pitch synchronous zero crossing peak amplitudes	Perceptual	Temporal	<input checked="" type="checkbox"/>	
Pulse clarity	Físico	Temporal		<input checked="" type="checkbox"/>
Pulse metric	Físico	Temporal		<input checked="" type="checkbox"/>
Rate-scale-frequency	Perceptual	Otro		
Relative Spectral-Perceptual Linear Prediction	Perceptual	Cepstral	<input checked="" type="checkbox"/>	
Rhythm Pattern	Perceptual	Temporal		<input checked="" type="checkbox"/>
Roughness feature	Perceptual	Frecuencial		
Sharpness	Perceptual	Frecuencial		<input checked="" type="checkbox"/>
Shimmer	Físico	Temporal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Short-time energy	Físico	Temporal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sparse coding tensor representation	Físico	Wavelet		<input checked="" type="checkbox"/>
Specific loudness sensation	Perceptual	Frecuencial	<input checked="" type="checkbox"/>	
Spectral center	Físico	Frecuencial		<input checked="" type="checkbox"/>
Spectral centroid	Físico	Frecuencial		<input checked="" type="checkbox"/>
Spectral crest factor	Físico	Frecuencial		<input checked="" type="checkbox"/>
Spectral decomposition	Físico	Wavelet		<input checked="" type="checkbox"/>
Spectral dispersion	Físico	Frecuencial		<input checked="" type="checkbox"/>
Spectral flatness	Físico	Frecuencial		<input checked="" type="checkbox"/>
Spectral flux	Físico	Frecuencial	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Spectral peaks	Físico	Frecuencial	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Spectral rolloff point	Físico	Frecuencial	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Spectral skewness and kurtosis	Físico	Frecuencial		<input checked="" type="checkbox"/>
Spectral slope	Físico	Frecuencial	<input checked="" type="checkbox"/>	
Spectrogram image features	Físico	Imagen	<input checked="" type="checkbox"/>	
Stabilized auditory image	Perceptual	Imagen	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Stereo panning spectrum feature:	Físico	Frecuencial		<input checked="" type="checkbox"/>
Subband energy ratio	Físico	Frecuencial	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Subband spectral flux	Físico	Frecuencial		<input checked="" type="checkbox"/>
Time-chroma images	Perceptual	Imagen		<input checked="" type="checkbox"/>
Volume	Físico	Temporal	<input checked="" type="checkbox"/>	
Wavelet-based direct approaches	Físico	Wavelet	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Zero-crossing peak amplitudes	Perceptual	Temporal	<input checked="" type="checkbox"/>	
Zero-crossing rate	Físico	Temporal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

ANEXO II: RESULTADOS COMPLETOS DE PRUEBAS

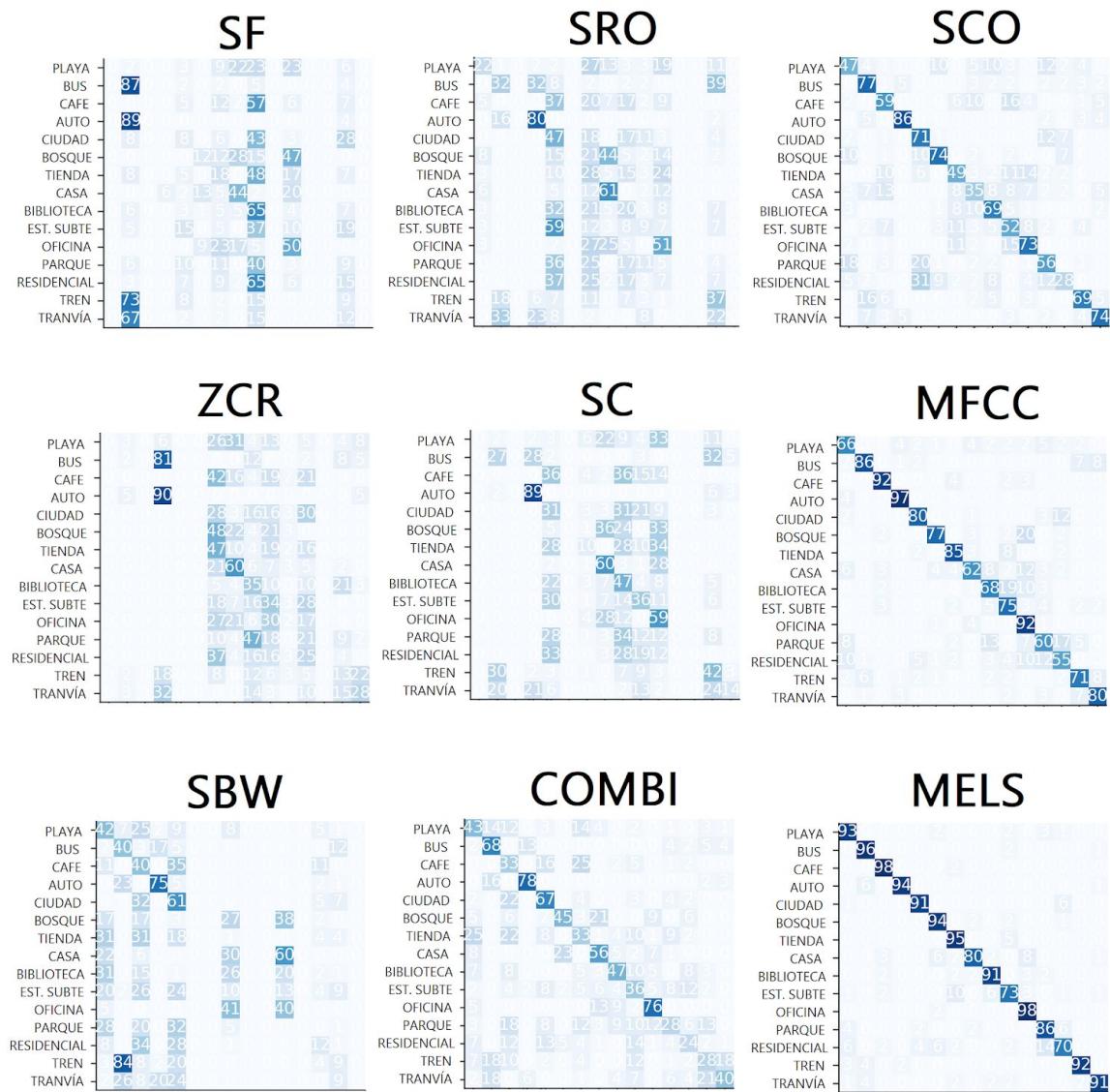
Tabla II.1: F-valor por clase para pruebas con descriptores utilizando DNN-0.

CLASES	SF	ZCR	SBW	SRO	SC	COMBI	SCO	MFCC	MELS
TUT-2017									
Playa	0.00	0.00	0.26	0.29	0.00	0.40	0.49	0.66	0.88
Autobús	0.37	0.03	0.28	0.32	0.30	0.59	0.70	0.89	0.92
Café	0.00	0.00	0.22	0.00	0.00	0.27	0.56	0.91	0.95
Auto	0.00	0.55	0.70	0.66	0.73	0.79	0.87	0.94	0.96
Ciudad	0.10	0.00	0.33	0.23	0.19	0.59	0.60	0.83	0.92
Bosque	0.17	0.00	0.00	0.00	0.00	0.51	0.74	0.83	0.90
Tiendas	0.17	0.22	0.00	0.16	0.14	0.31	0.50	0.85	0.90
Casa	0.40	0.42	0.24	0.46	0.44	0.55	0.40	0.69	0.87
Biblioteca	0.24	0.23	0.00	0.17	0.25	0.52	0.66	0.68	0.91
Est. Metro	0.00	0.20	0.00	0.11	0.29	0.34	0.48	0.70	0.79
Oficina	0.35	0.02	0.30	0.39	0.33	0.70	0.68	0.71	0.92
Parque	0.00	0.14	0.00	0.00	0.00	0.33	0.55	0.62	0.84
Residencial	0.00	0.00	0.16	0.00	0.00	0.28	0.38	0.57	0.75
Tren	0.08	0.15	0.11	0.31	0.35	0.32	0.73	0.73	0.95
Tranvía	0.00	0.33	0.00	0.00	0.23	0.47	0.75	0.81	0.93
GLOBAL	0.13	0.15	0.17	0.21	0.22	0.46	0.61	0.76	0.89
DESVÍO	0.15	0.17	0.19	0.20	0.21	0.16	0.14	0.11	0.06
TUT-2018									
Aeropuerto	0.31	0.32	0.31	0.30	0.34	0.41	0.58	0.45	0.64
Autobús	0.26	0.37	0.22	0.31	0.31	0.41	0.51	0.55	0.69
Metro	0.07	0.05	0.10	0.04	0.03	0.22	0.35	0.48	0.62
Est. Metro	0.00	0.24	0.25	0.00	0.00	0.36	0.49	0.32	0.52
Parque	0.18	0.24	0.14	0.21	0.20	0.29	0.40	0.39	0.57
Plaza	0.03	0.00	0.16	0.17	0.19	0.33	0.71	0.77	0.84
Tiendas	0.00	0.13	0.25	0.14	0.00	0.34	0.49	0.51	0.69
Peatonal	0.40	0.53	0.34	0.46	0.45	0.51	0.54	0.69	0.77
Tránsito	0.00	0.00	0.00	0.25	0.27	0.43	0.42	0.55	0.67
Tranvía	0.20	0.31	0.00	0.22	0.12	0.48	0.75	0.88	0.88
GLOBAL	0.15	0.22	0.18	0.21	0.19	0.38	0.52	0.56	0.69
DESVÍO	0.15	0.17	0.12	0.13	0.15	0.09	0.13	0.17	0.11

Tabla II.2: F-valor por clase para las pruebas con MFCC y Espectro Mel

CLASES	DNN & MFCC						DNN & MELS					
	A	B	C	D	E	F	A	B	C	D	E	F
TUT-2017												
Playa	0.43	0.66	0.8	0.55	0.72	0.75	0.8	0.86	0.93	0.67	0.85	0.86
Autobús	0.72	0.86	0.89	0.63	0.86	0.86	0.95	0.91	0.98	0.9	0.9	0.92
Café	0.78	0.9	0.92	0.79	0.9	0.87	0.94	0.91	0.92	0.86	0.96	0.95
Automóvil	0.82	0.93	0.94	0.76	0.89	0.86	0.96	0.97	0.97	0.91	0.97	0.95
Ciudad	0.78	0.85	0.87	0.84	0.82	0.84	0.91	0.87	0.88	0.89	0.85	0.87
Bosque	0.75	0.87	0.86	0.45	0.82	0.86	0.9	0.92	0.92	0.78	0.91	0.92
Tiendas	0.85	0.88	0.81	0.88	0.89	0.88	0.91	0.96	0.96	0.90	0.96	0.93
Casa	0.51	0.63	0.69	0.51	0.62	0.69	0.65	0.85	0.91	0.57	0.79	0.91
Biblioteca	0.58	0.76	0.77	0.55	0.74	0.76	0.87	0.86	0.92	0.82	0.88	0.88
Est. Metro	0.63	0.68	0.76	0.65	0.71	0.66	0.83	0.87	0.95	0.78	0.92	0.86
Oficina	0.76	0.83	0.84	0.66	0.86	0.84	0.86	0.89	0.95	0.8	0.90	0.95
Parque	0.45	0.79	0.84	0.44	0.77	0.73	0.70	0.82	0.82	0.61	0.77	0.80
Residencial	0.39	0.62	0.68	0.37	0.55	0.6	0.67	0.82	0.85	0.63	0.78	0.79
Tren	0.66	0.86	0.89	0.64	0.81	0.85	0.88	0.87	0.93	0.79	0.91	0.92
Tranvía	0.71	0.88	0.91	0.79	0.87	0.82	0.89	0.92	0.95	0.84	0.85	0.89
GLOBAL	0.65	0.80	0.83	0.63	0.79	0.79	0.85	0.89	0.92	0.78	0.88	0.89
DESVÍO	0.15	0.10	0.08	0.16	0.10	0.09	0.10	0.04	0.04	0.11	0.06	0.05
TUT-2018												
Aeropuerto	0.45	0.56	0.55	0.44	0.5	0.55	0.6	0.68	0.71	0.51	0.67	0.71
Autobús	0.58	0.6	0.61	0.56	0.61	0.58	0.66	0.72	0.75	0.66	0.68	0.73
Metro	0.46	0.5	0.49	0.39	0.51	0.51	0.56	0.66	0.71	0.59	0.64	0.65
Est. Metro	0.38	0.38	0.49	0.34	0.41	0.45	0.45	0.53	0.63	0.5	0.51	0.57
Parque	0.40	0.46	0.53	0.38	0.46	0.5	0.53	0.6	0.67	0.54	0.57	0.62
Plaza	0.78	0.76	0.8	0.74	0.77	0.77	0.78	0.81	0.84	0.81	0.82	0.8
Tiendas	0.51	0.57	0.6	0.5	0.53	0.53	0.64	0.70	0.76	0.67	0.68	0.70
Peatonal	0.68	0.68	0.70	0.67	0.67	0.66	0.76	0.8	0.82	0.75	0.79	0.78
Tránsito	0.57	0.62	0.62	0.56	0.61	0.65	0.62	0.67	0.73	0.63	0.63	0.67
Tranvía	0.84	0.86	0.89	0.85	0.87	0.87	0.90	0.90	0.91	0.87	0.91	0.92
GLOBAL	0.57	0.60	0.63	0.54	0.59	0.61	0.65	0.71	0.75	0.65	0.69	0.72
DESVÍO	0.16	0.14	0.13	0.17	0.14	0.13	0.13	0.11	0.08	0.13	0.12	0.10

TUT - 2017



TUT - 2018

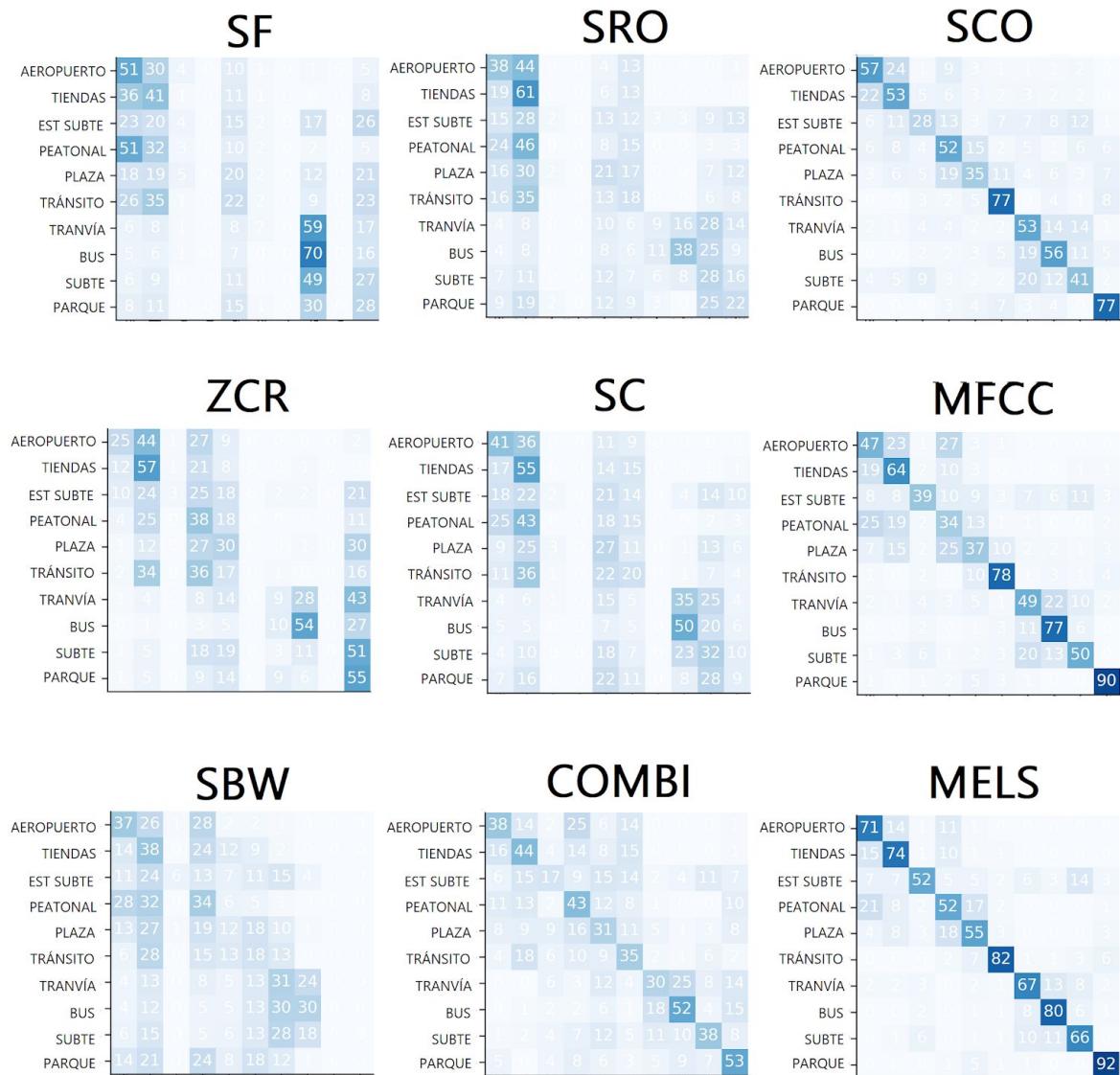


Figura II.2: Matrices de confusión para pruebas preliminares 1 sobre la base de datos TUT-2018.

Tabla II.3: F-valor por clase para pruebas con arquitecturas de CNN.

CLASES	CNN-A	CNN-B	CNN-C
TUT - 2017			
Playa	0.99	0.97	0.98
Autobús	0.98	0.95	0.96
Café	0.99	0.99	0.99
Automóvil	0.99	0.99	0.99
Ciudad	0.99	0.99	0.98
Bosque	0.99	0.98	0.99
Tiendas	0.99	0.99	0.99
Casa	0.99	0.99	0.98
Biblioteca	0.97	0.98	0.96
Est. Metro	0.98	0.97	0.97
Oficina	0.99	0.99	0.99
Parque	0.99	0.99	0.99
Residencial	0.98	0.98	0.98
Tren	0.99	0.98	0.98
Tranvía	0.98	0.97	0.98
GLOBAL	0.99	0.98	0.98
DESVÍO	0.01	0.01	0.01
TUT - 2018			
Aeropuerto	0.78	0.84	0.81
Autobús	0.80	0.79	0.87
Metro	0.71	0.76	0.80
Est. Metro	0.77	0.81	0.82
Parque	0.92	0.94	0.93
Plaza	0.68	0.76	0.77
Tiendas	0.79	0.84	0.84
Peatonal	0.68	0.75	0.75
Tránsito	0.89	0.94	0.87
Tranvía	0.81	0.81	0.84
GLOBAL	0.78	0.82	0.83
DESVÍO	0.08	0.07	0.05
TUT - COMBI			
Casa	0.97	0.98	0.99
Café	0.96	0.97	0.93
Tiendas	0.86	0.84	0.85
Parque	0.93	0.94	0.92
C. Urbana	0.81	0.79	0.79
C. Tránsito	0.92	0.88	0.86
Automóvil	0.86	0.79	0.88
Autobús	0.98	0.97	0.98
Tren	0.98	0.98	0.99
Metro	0.81	0.82	0.83
Est. Metro	0.80	0.79	0.82
GLOBAL	0.91	0.90	0.90
DESVÍO	0.07	0.08	0.07

Tabla II.4: F-valor por clase para pruebas con aumento de datos.

Base de datos	CLASES	ORIGINAL	MASCARAS AL.	FILTROS AL.
TUT-2017	Playa	0.98	0.99	0.98
	Autobús	0.96	0.97	0.97
	Café	0.99	0.99	0.99
	Automóvil	0.99	0.99	0.99
	Ciudad	0.98	0.98	0.98
	Bosque	0.99	0.99	0.99
	Tiendas	0.99	0.99	0.99
	Casa	0.98	0.99	0.97
	Biblioteca	0.96	0.98	0.99
	Est. Metro	0.97	0.98	0.97
	Oficina	0.99	0.98	0.99
	Parque	0.99	0.99	0.98
	Residencial	0.98	0.99	0.97
GLOBAL	Tren	0.98	0.99	0.99
	Tranvía	0.98	0.98	0.99
	GLOBAL	0.98	0.99	0.98
	DESVÍO	0.01	0.01	0.01
TUT-2018	Aeropuerto	0.77	0.89	0.84
	Autobús	0.85	0.91	0.88
	Metro	0.76	0.85	0.80
	Est. Metro	0.75	0.85	0.85
	Parque	0.92	0.94	0.93
	Plaza	0.69	0.80	0.73
	Tiendas	0.82	0.90	0.84
	Peatonal	0.68	0.81	0.73
	Tránsito	0.84	0.93	0.92
	Tranvía	0.77	0.89	0.89
	GLOBAL	0.79	0.88	0.84
	DESVÍO	0.07	0.05	0.07
	Casa	0.99	0.99	0.99
	Café	0.93	0.95	0.95
TUT-COMBI	Tiendas	0.85	0.90	0.90
	Parque	0.92	0.96	0.96
	C. Urbana	0.79	0.86	0.86
	C. Tránsito	0.86	0.95	0.94
	Automóvil	0.88	0.93	0.91
	Autobús	0.98	0.99	0.99
	Tren	0.99	0.99	0.99
	Metro	0.83	0.90	0.88
	Est. Metro	0.82	0.88	0.87
	GLOBAL	0.89	0.94	0.93
	DESVÍO	0.07	0.05	0.05

Tabla II.5: F-valor por clase utilizando diversos tipos de filtros convolucionales

CLASES	RECTANGULAR	TEMPORAL	FRECUENCIAL
Hogar	0.99	0.88	0.90
Café	0.86	0.71	0.76
Tiendas	0.85	0.68	0.72
Parque	0.94	0.86	0.87
Peatonal	0.79	0.69	0.74
Tránsito	0.90	0.81	0.82
Autobús	0.89	0.65	0.82
Automóvil	0.99	0.89	0.97
Tren	0.90	0.70	0.85
Metro	0.84	0.56	0.78
Est. Metro	0.82	0.56	0.70
GLOBAL	0.89	0.73	0.81
DESVÍO	0.06	0.12	0.08

Tabla II.6: F-valor por clase con diversas técnicas de entrenamiento utilizando BA-DB.

CLASES	NINGUNO	MÁSCARAS	FILTROS	RECORTE	TODOS
Casa	0.61	0.70	0.77	0.49	0.56
Café	0.18	0.23	0.62	0.11	0.61
Tiendas	0.50	0.53	0.58	0.45	0.64
Parque	0.64	0.54	0.66	0.52	0.45
Peatonal	0.41	0.43	0.49	0.05	0.17
Tránsito	0.60	0.43	0.59	0.07	0.18
Autobús	0.55	0.55	0.58	0.44	0.40
Automóvil	0.79	0.65	0.80	0.67	0.58
Tren	0.10	0.03	0.14	0.05	0.21
Metro	0.24	0.22	0.30	0.17	0.22
Est. Metro	0.25	0.30	0.39	0.17	0.25
GLOBAL	0.44	0.42	0.54	0.29	0.39
DESVÍO	0.22	0.20	0.20	0.23	0.19

Tabla II.7: Resultados de métricas por clase para el modelo final utilizando BA-DB..

CLASES	Precisión	Sensibilidad	F-valor (F1)	Muestras
Casa	0.66	0.94	0.77	807
Café	0.61	0.63	0.62	819
Tiendas	0.65	0.52	0.58	622
Parque	0.71	0.63	0.66	608
Peatonal	0.73	0.37	0.49	556
Tránsito	0.94	0.43	0.59	566
Autobús	0.49	0.72	0.58	750
Automóvil	0.78	0.83	0.80	825
Tren	0.35	0.08	0.14	662
Metro	0.21	0.54	0.30	453
Est. Metro	0.51	0.31	0.39	497
PROMEDIO	0.61	0.57	0.56	651
DESVÍO	0.20	0.24	0.20	132

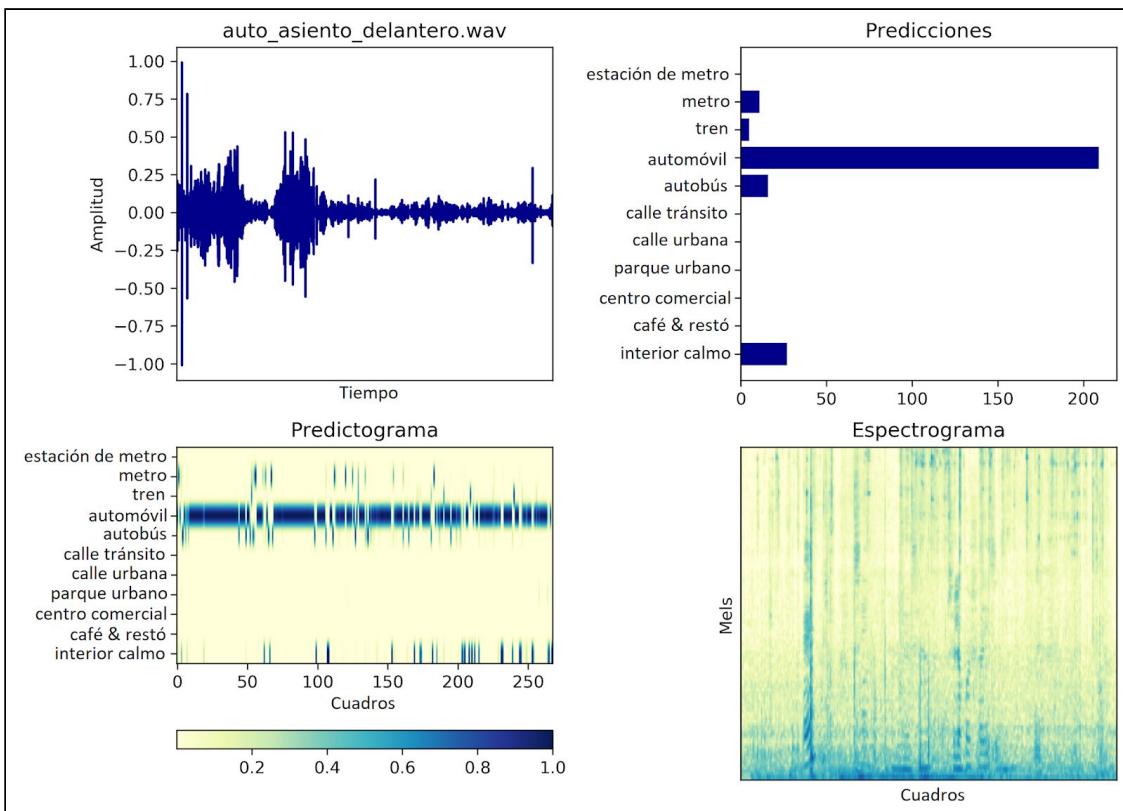


Figura II.3: Análisis de predicciones del modelo final sobre escena de automóvil de BA-DB.

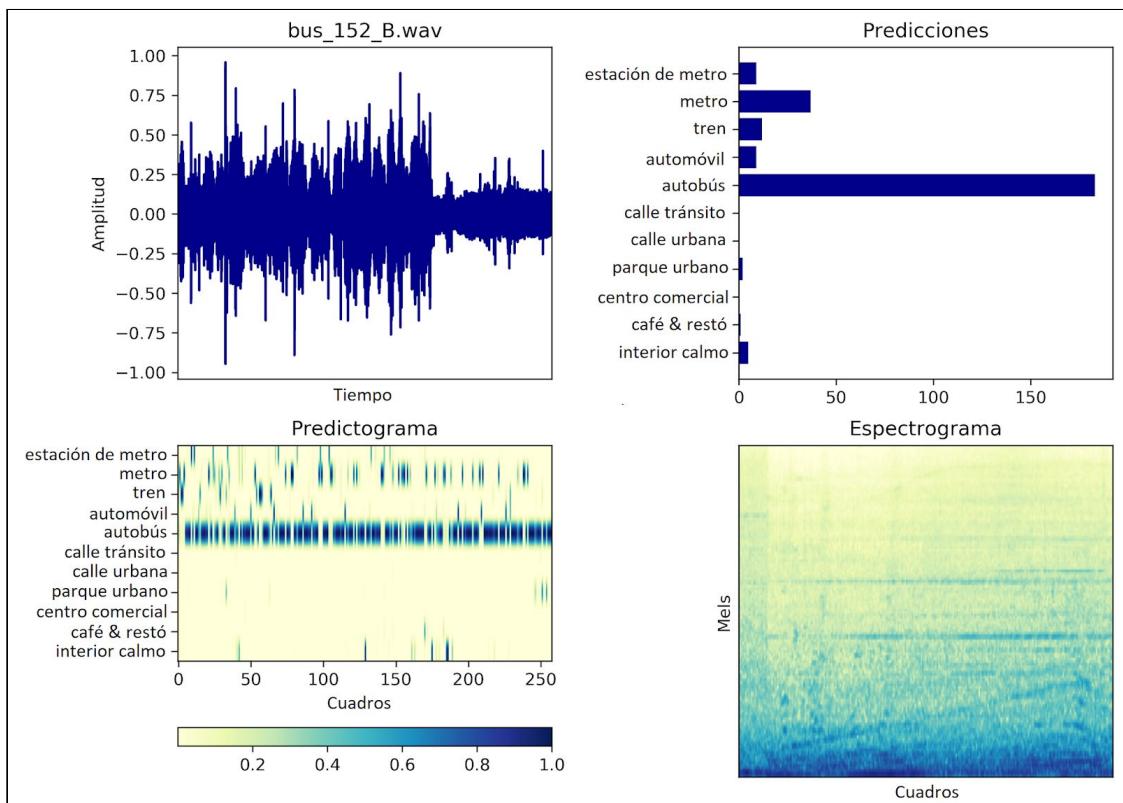


Figura II.4: Análisis de predicciones del modelo final sobre escena de autobús de BA-DB

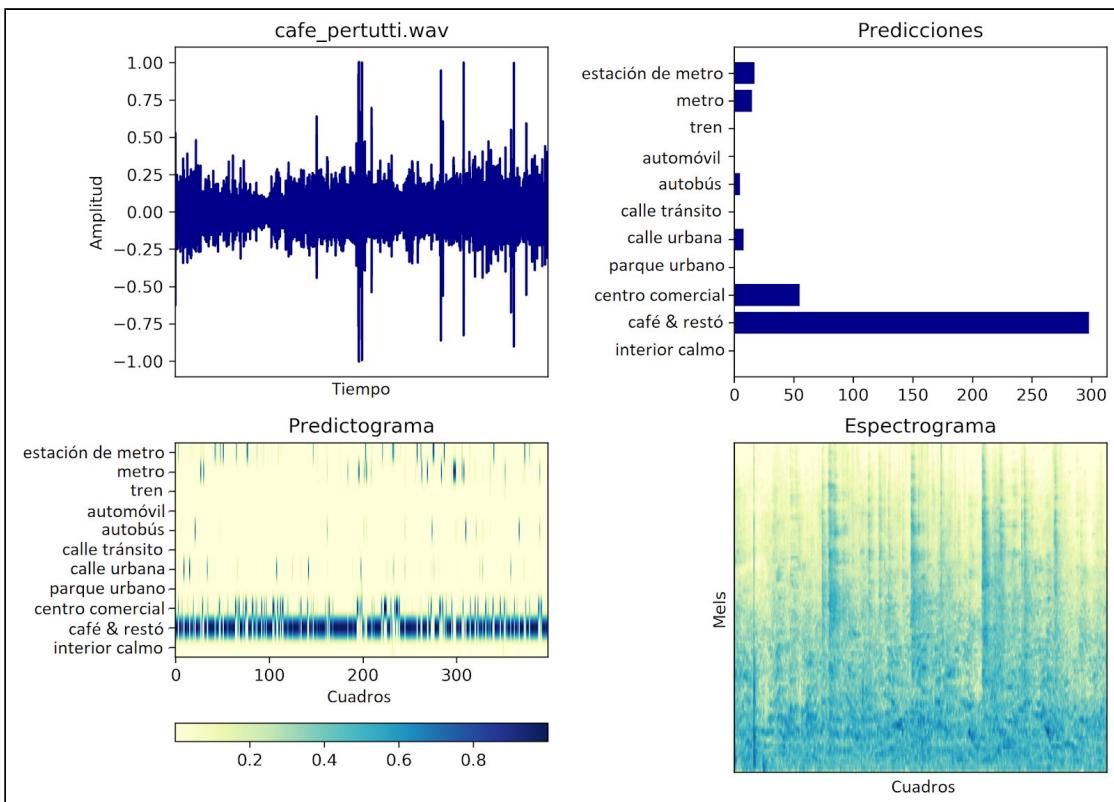


Figura II.5: Análisis de predicciones del modelo final sobre escena de café de BA-DB

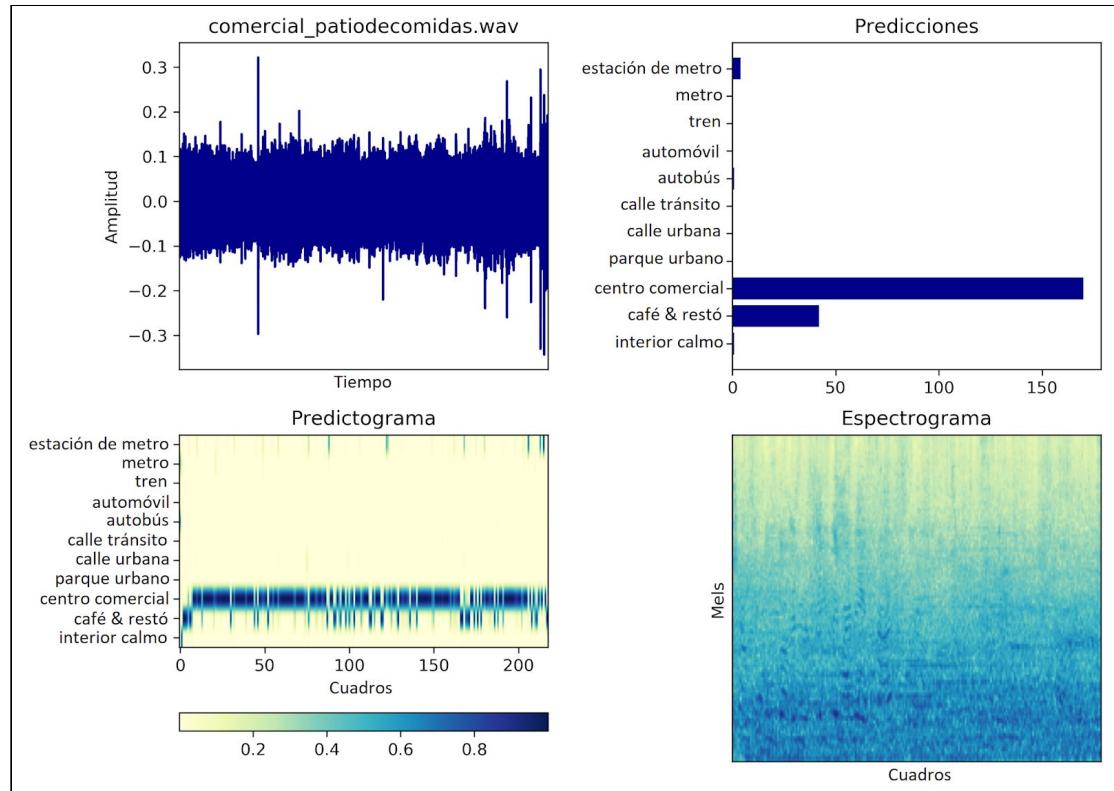


Figura II.6: Análisis de predicciones del modelo final sobre escena de patio de comidas de BA-DB.

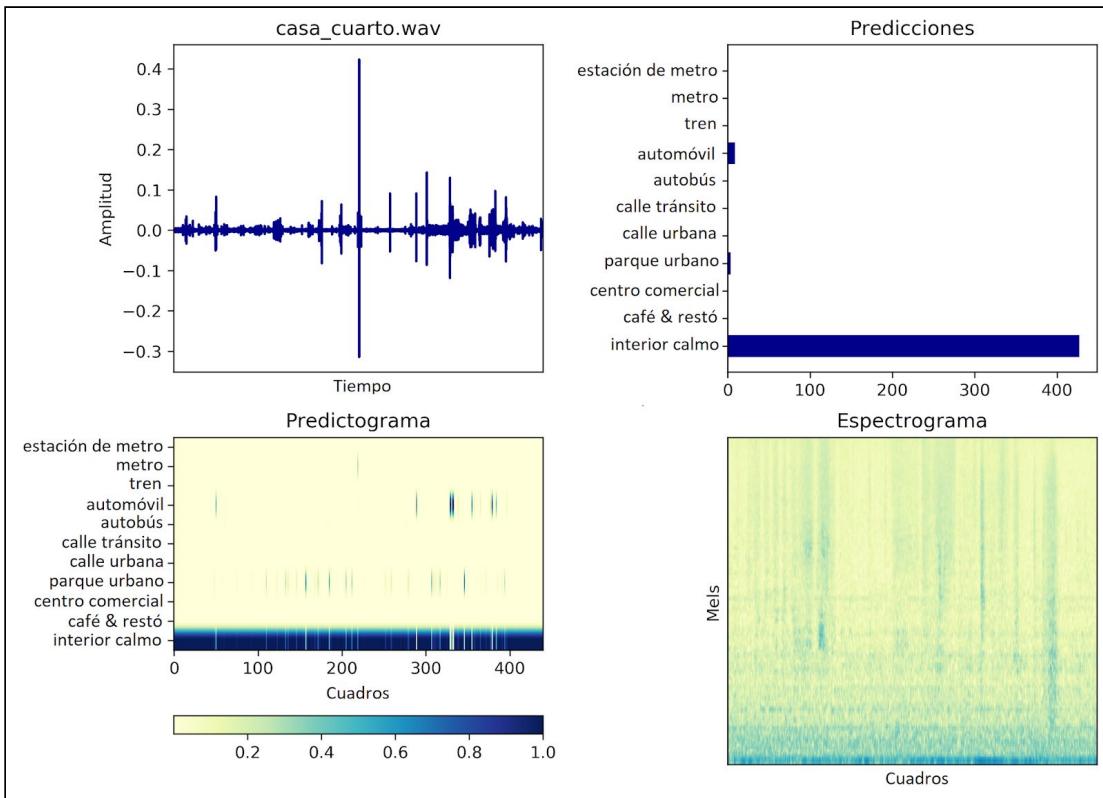


Figura II.7: Análisis de predicciones del modelo final sobre escena de habitación de BA-DB.

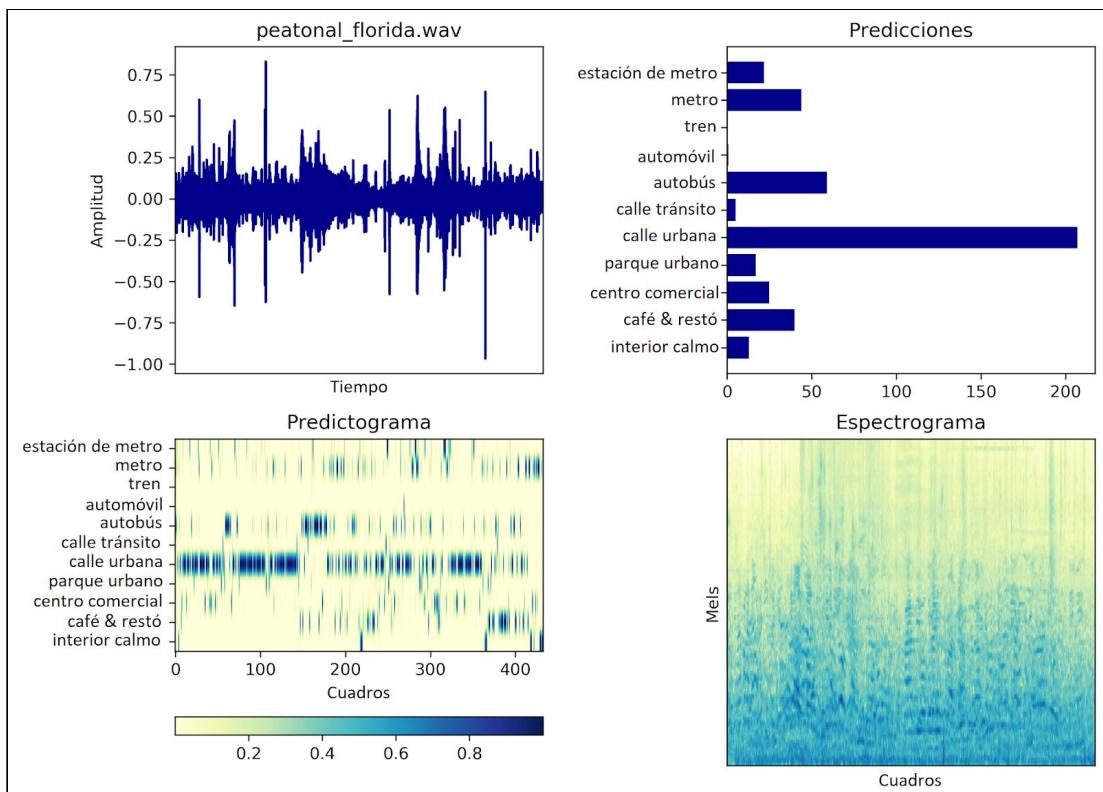


Figura II.8: Análisis de predicciones del modelo final sobre escena de calle urbana de BA-DB

ANEXO III: MODELO FINAL IMPLEMENTADO EN ANDROID

Tabla III.1: Detalle de arquitectura de red neuronal convolucional implementada.

Bloque	Tipo de capa	X	Y	Z	Parámetros
1	Normalización de Lote	128	128	1	4
	Capa Convolutacional (3,3)	128	128	24	240
	Activación (Re-LU)	128	128	24	0
2	Agrupamiento de máximos (2,2)	64	64	24	0
	Normalización de Lote	64	64	24	96
	Capa Convolutacional (3,3)	64	64	24	5208
3	Activación (Re-LU)	64	64	24	0
	Agrupamiento de máximos (2,2)	32	32	24	0
	Normalización de Lote	32	32	24	96
4	Capa Convolutacional (3,3)	32	32	48	10416
	Activación (Re-LU)	32	32	48	0
	Agrupamiento de máximos (2,2)	16	16	48	0
5	Normalización de Lote	16	16	48	192
	Capa Convolutacional (2, 2)	16	16	48	9264
	Activación (Re-LU)	16	16	48	0
6	Agrupamiento de máximos (2,2)	8	8	48	0
	Normalización de Lote	8	8	48	192
	Capa Convolutacional (2, 2)	8	8	48	9264
total	Activación (Re-LU)	8	8	48	0
	Agrupamiento de máximos (2,2)	4	4	48	192
	Normalización de Lote	4	4	48	192
6	Achatamiento	0	0	768	0
	Capa Densa (Activación Softmax)	0	0	11	8459
					43623

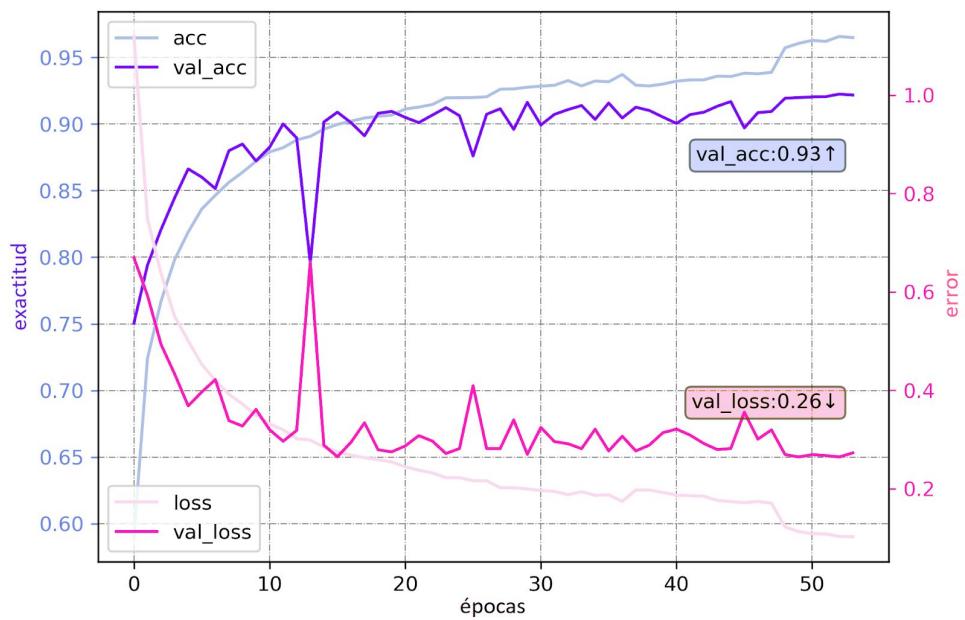


Figura III.1: Progreso de entrenamiento automático de modelo final implementado.

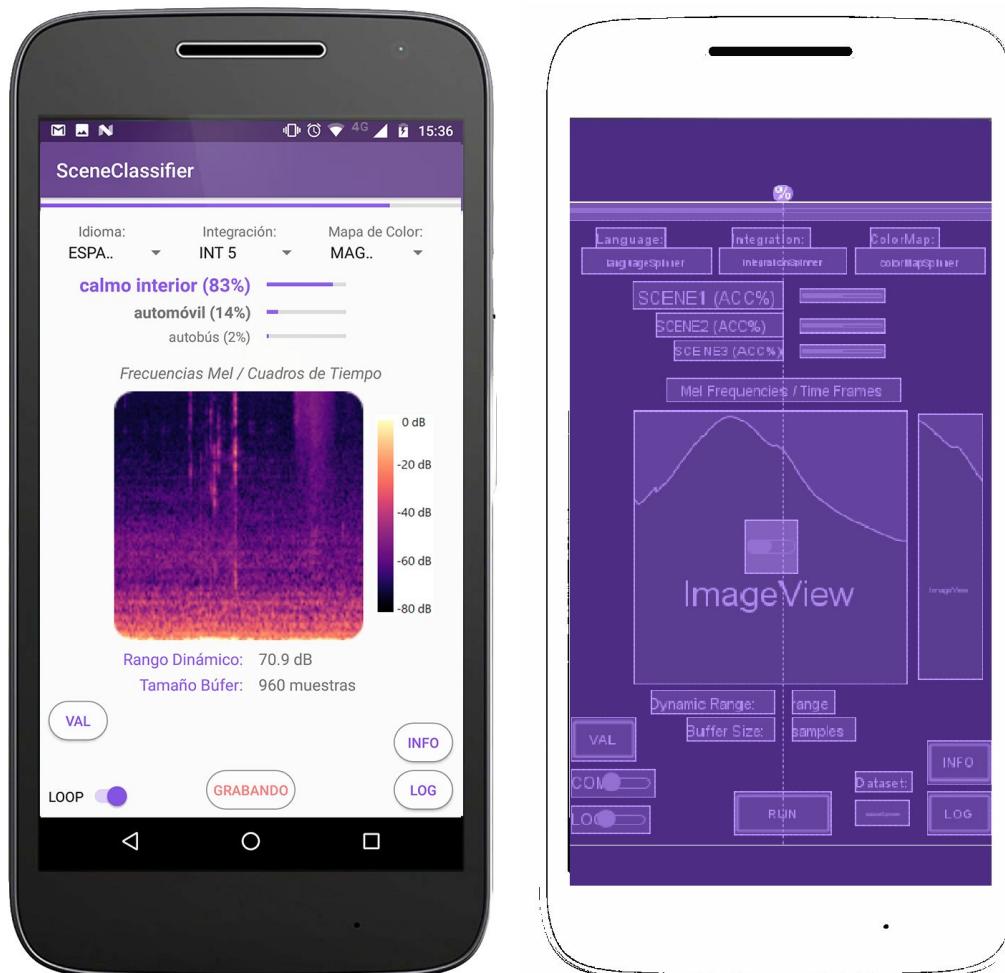


Figura III.2: Interfaz de usuario de la aplicación desarrollada en Android.

ANEXO IV: CÓDIGO IMPLEMENTADO

```

def zero_crossing_rate(audio):
    return np.mean(librosa.feature.zero_crossing_rate(y=audio, frame_length=2048, hop_length=1024, center=True),axis=1)
def spectrum(audio):
    return np.abs(stft(audio))
def spectral_centroid(S):
    return np.mean(librosa.feature.spectral_centroid(S=S,sr=48000, n_fft=2048, hop_length=1024),axis=1)
def spectral_bandwidth(S):
    return np.mean(librosa.feature.spectral_bandwidth(S=S,sr=48000, n_fft=2048, hop_length=1024, p=2),axis=1)
def spectral_contrast(S):
    return np.mean(librosa.feature.spectral_contrast(S=S,sr=48000, n_fft=2048, hop_length=1024,n_bands=6),axis=1)
def spectral_flatness(S):
    return np.mean(librosa.feature.spectral_flatness(S=S, n_fft=2048, hop_length=1024, amin=1e-10, power=2.0),axis=1)
def spectral_rolloff(S):
    return np.mean(librosa.feature.spectral_rolloff(S=S,sr=48000, n_fft=2048, hop_length=1024),axis=1)
def rms(S):
    return np.mean(librosa.feature.rms(S=S,frame_length=2048,hop_length=1024))
def mfcc(S,mels=50):
    return np.mean(librosa.feature.mfcc(S=mel_spectrogram(S=mels,mels=mels),sr=48000,n_mfcc=mels,dct_type=2),axis=1)
def melspectrogram(audio,ref=np.max,sr=48000,n_mels=128,n_fft=2048,hop_length=1024,fmax=20000):
    S = librosa.feature.melspectrogram(y=audio,sr=sr,n_mels=n_mels,n_fft=n_fft,hop_length=hop_length,fmax=fmax)
    spectrogram = np.flipud(librosa.power_to_db(S, ref=ref))
    return spectrogram[:,1:-1]

```

Figura IV.1: PYTHON: funciones definidas para extracción de descriptores de escenas sonoras.

```

import keras
import tools
tools.dataset_feature_extraction(
    source_dir      = "J:\\DataOffline\\Datasets\\SOUND_SCENES_AUDIO_HPF",
    destination_dir = "J:\\DataOffline\\Datasets\\SOUND_SCENES_MELSPECTROGRAM",
    feature_type    ='mel_specs'
)
finalCombi = tools.Combination(
    source_path     = "J:\\DataOffline\\Datasets\\SOUND_SCENES_MELSPECTROGRAM",
    destination_path = "J:\\DataOffline\\Datasets\\COMBINATIONS\\COMBI_FINAL_MELSPECTROGRAM"
)
finalCombi.add("00_quiet_interior", ["tut17_07_home","tut17_08_library","tut17_10_office"])
finalCombi.add("01_cafe_resto", ["tut17_02_cafe"])
finalCombi.add("02_mall", ["tut18_06_shopping_mall"])
finalCombi.add("03_urban_park", ["tut18_04_park","tut18_05_public_square","tut17_11_park"])
finalCombi.add("04_street_urban", ["tut18_07_street_pedestrian"])
finalCombi.add("05_street_traffic", ["tut18_08_street_traffic","tut17_04_city_center"])
finalCombi.add("06_bus", ["tut18_01_bus","tut17_01_bus"])
finalCombi.add("07_car", ["tut17_03_car"])
finalCombi.add("08_train", ["tut17_13_train"])
finalCombi.add("09_metro", ["tut18_03_metro"])
finalCombi.add("10_metro_station", ["tut18_02_metro_station","tut17_09_metro_station"])
finalCombi.display_support()
finalCombi.generate()
train_dataset = tools.Dataset(
    name          = "COMBI_FINAL",
    features_dir  = "J:\\DataOffline\\Datasets\\COMBINATIONS\\COMBI_FINAL_MELSPECTROGRAM",
    xy_dir        = "J:\\DataOffline\\Datasets\\TEMP_TRAIN",
)
train_dataset.purge()
train_dataset.plot_support()
train_dataset.compile_XY(split=True,val_size=0.2,test_size=0.2)
X_train, Y_train, X_val, Y_val, X_test, Y_test = train_dataset.load_XY(is_split=True,normalize=True)

```

Figura IV.2: PYTHON: combinación de base de datos para el entrenamiento del modelo.

```

classifier = tools.Classifier(    name    = "butter_DA",      labels = train_dataset.labels)
classifier.model = keras.models.Sequential()
# LAYER 1:
classifier.model.add(keras.layers.BatchNormalization(input_shape=train_dataset.input_shape))
classifier.model.add(keras.layers.Conv2D(filters=24, kernel_size=(3, 3), padding="same"))
classifier.model.add(keras.layers.Activation('relu'))
classifier.model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
# LAYER 2:
classifier.model.add(keras.layers.BatchNormalization())
classifier.model.add(keras.layers.Conv2D(filters=24, kernel_size=(3, 3), padding="same"))
classifier.model.add(keras.layers.Activation('relu'))
classifier.model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
# LAYER 3:
classifier.model.add(keras.layers.BatchNormalization())
classifier.model.add(keras.layers.Conv2D(filters=48, kernel_size=(3, 3), padding="same"))
classifier.model.add(keras.layers.Activation('relu'))
classifier.model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
# LAYER 4:
classifier.model.add(keras.layers.BatchNormalization())
classifier.model.add(keras.layers.Conv2D(filters=48, kernel_size=(2, 2), padding="same"))
classifier.model.add(keras.layers.Activation('relu'))
classifier.model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
# LAYER 5:
classifier.model.add(keras.layers.BatchNormalization())
classifier.model.add(keras.layers.Conv2D(filters=48, kernel_size=(2, 2), padding="same"))
classifier.model.add(keras.layers.Activation('relu'))
classifier.model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
# LAYER 6:
classifier.model.add(keras.layers.BatchNormalization())
classifier.model.add(keras.layers.Flatten())
# LAYER 7:
classifier.model.add(keras.layers.Dense(activation="softmax", units=train_dataset.num_classes))
# BUILD MODEL
classifier.model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
classifier.model.summary()
BATCH_SIZE = 32
N_EPOCHS = 20
training_generator = tools.DataGenerator(X_train,Y_train,BATCH_SIZE,spec_filter=True, spec_drop=False)
validation_generator = tools.DataGenerator(X_val,Y_val,BATCH_SIZE,spec_filter=False, spec_drop=False)
classifier.model.fit_generator(
    generator          = training_generator,
    epochs             = N_EPOCHS,
    callbacks          = classifier.set_callbacks(),
    validation_data    = validation_generator,
    class_weight       = train_dataset.class_weights,
    shuffle            = False,
)
classifier.plot_learning()
classifier.save_as_keras_model()
classifier.save_as_tflite(keras model path="models\\CNN final DA butter 92.hdf5")

```

Figura IV.3: PYTHON: arquitectura de red neuronal para clasificador de escenas sonoras.

```

def spec_mask(data,low_bin,high_bin,null=0,flip=False):
    if flip : data = np.array([np.flipud(spec) for spec in data])
    count,bin_size,frame_size,n_channels=np.shape(data)
    data[:,0:low_bin,:,:] = null
    data[:,high_bin:bin_size,:,:] = null
    if flip : data = np.array([np.flipud(spec) for spec in data])
    return data

def spec_dropout(X_batch,freq_strips=True,time_strips=True,n_strips=1,min_strip_size=1,tolerance=0.2,null_value=0):
    batch_size, bin_size, frame_size, n_channels = np.shape(X_batch)
    n_samples_to_augment = np.random.randint(low=int(0.25*batch_size),high=int(0.75*batch_size))
    augment_selection = random.sample(list(np.arange(batch_size)),n_samples_to_augment)
    limit = np.minimum(bin_size,frame_size)
    for _ in range(n_strips):
        strip_sizes = list(np.random.randint(low=min_strip_size,high=0.2*limit/2,size=n_samples_to_augment))
        strip_centers = [int(np.random.uniform(strip_size,limit-strip_size)) for strip_size in strip_sizes]
        strips_selection = [list(np.arange(center-size:center+size+1)) for size,center in zip(strip_sizes,strip_centers)]
        for i in range(n_samples_to_augment):
            if freq_strips: X_batch[augment_selection[i],strips_selection[i]] = null_value
            if time_strips: X_batch[augment_selection[i],:,strips_selection[i]] = null_value
    return X_batch

def spec_filters(X_batch):
    batch_size, bin_size, frame_size, n_channels = np.shape(X_batch)
    X_batch = np.reshape(X_batch,newshape=(batch_size,bin_size,frame_size))
    batch_filters = scale_filters(butterworth_filters(bin_size=bin_size,n_filters=batch_size),n_mels=bin_size)
    batch_matrices = [np.tile(batch_filter,(bin_size,1)).transpose() for batch_filter in batch_filters]
    filtered_batch = [x*np.flipud(matrix) for x,matrix in zip(X_batch,batch_matrices)]
    return np.reshape(filtered_batch,newshape=(batch_size, bin_size, frame_size,n_channels))

def butterworth_filters(bin_size,n_filters=1,wc_lims=[0.4,1],n_lims=[1,20]):
    w_norm = np.linspace(0,1,bin_size)
    butterworth_filters = []
    while n_filters>0:
        wc = np.random.uniform(*wc_lims)
        n = np.random.randint(*n_lims)
        butterworth_lpf = [1/np.sqrt(1+(w/wc)**(2*n)) for w in w_norm]
        butterworth_hpf = np.flip(butterworth_lpf)
        butterworth_bpf = butterworth_lpf * butterworth_hpf
        butterworth_filters.append(butterworth_bpf)
        n_filters-=1
    return butterworth_filters

def scale_filters(filter_arrays,n_mels,f_max=20000):
    mel_freqs = librosa.core.mel_frequencies(n_mels=n_mels, fmin=0.0, fmax=f_max, htk=False)
    mel_indeces = [int(np.floor(freq*n_mels/f_max)) for freq in mel_freqs]
    filtered_arrays = []
    for filter_array in filter_arrays:
        filtered_array = np.flip([filter_array[index] for index in mel_indeces]) # mel_scale
        filtered_array = scipy.signal.savgol_filter(filtered_array,window_length=9,polyorder=3) # smooth
        while np.any(filtered_array <=0): filtered_array += 0.01
        filtered_array = (20 * np.log10(filtered_array) + 80)/80 # convert to dB & normalize
        filtered_arrays.append(filtered_array)
    return filtered_arrays

```

Figura IV.4: PYTHON: procesos de aumento de datos definidos para lote de espectrogramas.

```

public class MainActivity extends AppCompatActivity implements AdapterView.OnItemSelectedListener {

    // INITIALIZE GLOBAL VARIABLES:
    public int sampleRate=        48000;
    public int hopLength =        1024;
    public int fftSize =          2048;
    public int melFreqs =         128;
    public int frames =           128;
    public int nInteg =           5;
    public int numClasses;
    public int sizeInFloats;
    public String saveName;
    public int targetSamples = (frames+1)*hopLength;
    public int totalSamples = sampleRate* 4;
    public int accuracy_0=0, accuracy_1=0, accuracy_2=0;
    public float[] pred_0, pred_1, pred_2, pred_3, pred_4;
    public float[] audioSignal = new float[targetSamples];
    public float[][] spectrogram = new float[melFreqs][frames];
    private static final int STORAGE_CODE =      1;
    public static final int RECORD_AUDIO_CODE = 200;
    public boolean loopRunning = true;
    public String[] labels;
    public String modelFileName, dynamicRange, label_0="...", label_1="...", label_2="...";
}

```

Figura IV.5: JAVA: Inicialización de variables globales para la aplicación Android desarrollada.

```

int source = MediaRecorder.AudioSource.UNPROCESSED;
int channel = AudioFormat.CHANNEL_IN_MONO;
int format = AudioFormat.ENCODING_PCM_FLOAT;
int readMode = AudioRecord.READ_BLOCKING;
boolean isRecording;
int minBuffer = AudioRecord.getMinBufferSize(sampleRate, channel, format);
sizeInFloats = minBuffer / 4; // FLOAT32 --> 32 bits --> 4 bytes
float[] audioMiniBuffer = new float[sizeInFloats]; // small temp buffer
float[] recordedAudio = new float[totalSamples]; // 3 sec of samples
float[] targetAudio = new float[targetSamples]; // target signal
AudioRecord recorder = new AudioRecord(source, sampleRate, channel, format, minBuffer);
if (recorder.getState() == 0) { Log.d("console", " ERROR : AudioRecord uninitialized");}
else if (recorder.getState() == 1) { Log.d("console", "REC: Recording!...");}
    recorder.startRecording();
    isRecording = true;
    int offset = 0;
    int readCounter;
    int sampleIndex = 0;
    int totalRead = 0;
    while (isRecording) {
        readCounter = recorder.read(audioMiniBuffer, offset, sizeInFloats, readMode);
        totalRead += readCounter;
        if (AudioRecord.ERROR_INVALID_OPERATION != readCounter) { // error check.
            for (int i = 0; i < sizeInFloats; i++) {
                if (sampleIndex < totalSamples) {
                    recordedAudio[sampleIndex] = audioMiniBuffer[i];
                }
                if (sampleIndex >= totalSamples) {
                    if (recorder.getState() == 1) {
                        recorder.stop();
                        recorder.release();
                        isRecording = false;
                        break;
                    }
                    sampleIndex += 1;
                }
            }
        }
        int sourcePos = totalSamples - targetSamples;
        System.arraycopy(recordedAudio, sourcePos, targetAudio, 0, targetSamples);
        targetAudio = normalize(targetAudio);
        float totalDuration = ((float)(targetAudio.length))/sampleRate;
    }
    return targetAudio;
}

```

Figura IV.6: JAVA: extracto para la adquisición de muestras de audio en aplicación de Android.

```

private class ProcessTask extends AsyncTask<float[], Integer, float[][]> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        Log.d("console", "PROCESS: started");
        loadingBar.setVisibility(View.VISIBLE);
    }
    @Override
    protected float[][] doInBackground(float[]... signals) {
        publishProgress(0);
        float[] signal = signals[0];
        float[][] melSpec = new MelSpectrogram( signal, sampleRate, melFreqs, frames, fftSize,hopLength)
        melSpec.getSpectrogram();
        publishProgress(100);
        return melSpec;
    }
    @Override
    protected void onProgressUpdate(Integer... progress) {
        super.onProgressUpdate(progress);
        horizontalProgressBar.setProgress(progress[0]);
    }
    @Override
    protected void onPostExecute(float[][] melSpec) {
        super.onPostExecute(melSpec);
        System.arraycopy(melSpec,0,spectrogram,0,melSpec.length);
        showSpectrogram();
        InferenceTask inferenceTask = new InferenceTask();
        inferenceTask.execute(melSpec);
    }
}

```

Figura IV.7: JAVA: hilo de procesamiento dedicado a la extracción de spectrogramas.

```

private class InferenceTask extends AsyncTask<float[][][], Integer, float[][]> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
    @Override
    protected float[][][] doInBackground(float[][][] specs) {
        publishProgress(0);
        float[][] spec = specs[0];
        float[][][] inputTensor = new float[1][melFreqs][frames][1];
        for (int i = 0; i < frames; i++){
            for (int j = 0; j < melFreqs; j++) {
                inputTensor[0][i][j][0] = spec[i][j];
            }
        }
        float[][] outputTensor = new float[1][numClasses];
        try {
            Interpreter tensorFlowLite = new Interpreter(loadModelFile());
            tensorFlowLite.run(inputTensor,outputTensor);
            tensorFlowLite.close();
            publishProgress(100);
        } catch (Exception ex){ex.printStackTrace();}
        return outputTensor;
    }
    @Override
    protected void onProgressUpdate(Integer... progress) {
        super.onProgressUpdate(progress);
        horizontalProgressBar.setProgress(progress[0]);
    }
    @Override
    protected void onPostExecute(float[][][] outputTensor) {
        super.onPostExecute(outputTensor);
        savePredictions(outputTensor);
        showUI();
        if (loopRunning) {executeTasks();}
    }
}

```

Figura IV.8: JAVA: hilo de procesamiento dedicado a la inferencia de red neuronal.

```

class MelSpectrogram {
    private int sampleRate;
    private int nMels;
    private int frames;
    private int fftSize;
    private int hopLength;
    private float[][] melSpectrogram;
    MelSpectrogram(float[] signal, int sampleRate, int nMels, int frames, int fftSize, int hopLength){
        this.sampleRate = sampleRate;
        this.nMels = nMels;
        this.frames = frames;
        this.fftSize = fftSize;
        this.hopLength=hopLength;
        float[][] linSpec = linearSpectrogram(signal);
        float[][] melFilters = melFilterBank();
        float[][] filteredSpectrogram = dot(melFilters, linSpec);
        float[][] melSpectrogram_dB = power_to_db(filteredSpectrogram);
        float[][] melSpectrogram_dB_flip = flipUD(melSpectrogram_dB);
        melSpectrogram = normalize(melSpectrogram_dB_flip); }
    float[][] getSpectrogram(){ return melSpectrogram; }
}

```

Figura IV.9: JAVA: inicialización de clase para la extracción de espectrogramas en Android.

```

private float[] hanningWindow(int size){
    float[] window = new float[size];
    int m = size / 2;
    for (int n = -m; n < m; n++)
        window[m + n] = 0.5f + 0.5f * (float)(Math.cos(n * Math.PI / (m + 1)));
    return window;
}

private float[] fft(float[] signal){
    float[] spectrum = new float[fftSize / 2 + 1];
    int[] reverse;
    reverse = new int[fftSize];
    reverse[0] = 0;
    for (int limit = 1, bit = fftSize / 2; limit < fftSize; limit <= 1, bit >= 1)
        for (int i = 0; i < limit; i++)
            reverse[i + limit] = reverse[i] + bit;
    //get real and imaginary arrays:
    float[] real = new float[fftSize];
    float[] imag = new float[fftSize];
    for (int i = 0; i < fftSize; ++i){
        real[i] = signal[reverse[i]];
        imag[i] = 0.0f;
    }
    for (int halfSize = 1; halfSize < real.length; halfSize *= 2){
        float k = -(float) Math.PI/halfSize;
        float phaseShiftStepR = (float) Math.cos(k);
        float phaseShiftStepI = (float) Math.sin(k);
        // current phase shift
        float phaseShiftR = 1.0f;
        float phaseShiftI = 0.0f;
        for (int fftStep = 0; fftStep < halfSize; fftStep++){
            for (int i = fftStep; i < real.length; i += 2 * halfSize){
                int off = i + halfSize;
                float tr = (phaseShiftR * real[off]) - (phaseShiftI * imag[off]);
                float ti = (phaseShiftR * imag[off]) + (phaseShiftI * real[off]);
                real[off] = real[i] - tr;
                imag[off] = imag[i] - ti;
                real[i] += tr;
                imag[i] += ti;
            }
            float tmpR = phaseShiftR;
            phaseShiftR = (tmpR * phaseShiftStepR) - (phaseShiftI * phaseShiftStepI);
            phaseShiftI = (tmpR * phaseShiftStepI) + (phaseShiftI * phaseShiftStepR);}
        for (int i = 0; i < spectrum.length; i++)
            spectrum[i] = (float) Math.sqrt(real[i] * real[i] + imag[i] * imag[i]);}
    return spectrum;
}

```

Figura IV.10: JAVA: funciones para el cómputo de transformada de Fourier en Android.

```

private float[][] linearSpectrogram(float[] signal){
    float[] window = hanningWindow(fftSize); // Licensed under the Apache License
    // COMPUTE FFTs
    int fftFREQS = 1+fftSize/2;
    float[] fftBuffer = new float[fftSize];
    float[][] linSpectrogram = new float[fftFREQS][frames];
    int startSample = 0; // initialize sample counter
    for (int column=0;column<frames;column++) {
        for (int i=0; i<fftSize; i++) {fftBuffer[i] = signal[startSample+i] * window[i];}
        float[] fftOutput = fft(fftBuffer); // compute FFT
        for (int k=0;k<fftFREQS;k++) {linSpectrogram[k][column] = fftOutput[k];}
        startSample += hopLength; // increment hop
    }
    float[][] linearPowerSpectrogram = new float[fftFREQS][frames];
    for (int j=0 ; j < frames ; j++ ) {
        for (int i=0 ; i < fftFREQS ; i++ ) {
            linearPowerSpectrogram[i][j] = (float) Math.pow(Math.abs(linSpectrogram[i][j]),2);}
    }
    return linearPowerSpectrogram;
}

private float[][] melFilterBank(){
    int fMax = 2000;
    int nFREQS = 1 + fftSize / 2;
    // Fill in the linear scale *(below 1KHz);
    float max_mel = (float)
        ((1000.0)/(200.0f / 3.0f) + Math.log(fMax/1000.0f)/(Math.log(6.4f)/27.0f));
    float min_mel = 0.0f;
    float[] mels = linSpace(min_mel, max_mel, nMels+2);
    float[] freqs = new float[nMels+2];
    for (int i=0; i<nMels+2; i++) { freqs[i] = (200.0f/3.0f) * mels[i];}
    // Fill the nonlinear scale *(above 1KHz) ;
    for (int i=0; i<nMels+2; i++) {
        if (mels[i] >= (1000) / (200.0 / 3)) {
            freqs[i] =(float)
                (1000.0*Math.exp((Math.log(6.4)/27.0)*(mels[i]-(1000.0)/(200.0/3.0))));}
    }
    // Triangular Filter: Center freqs of each FFT bin;
    float[] fftFREQS = linSpace(0, ((float)sampleRate) / 2, nFREQS);
    float[][] ramps = new float[nMels+2][nFREQS];
    for (int i=0; i<nMels+2; i++) {
        for (int j=0; j < nFREQS; j++) {
            ramps[i][j]= freqs[i] - fftFREQS[j];}}
    // lower and upper slopes for all bins. ;
    float[] fdiff = new float[nMels+1];
    for (int i=0; i<nMels+1; i++ ) { fdiff[i] = freqs[i+1]-freqs[i]; }
    // Get the weights;
    float[][] weights = new float[nMels][nFREQS];
    for (int m=0; m < nMels; m++ ) {
        for (int k=0; k < nFREQS; k++) {
            weights[m][k] = (float) Math.max(
                0, Math.min((-1.0)*ramps[m][k]/fdiff[m],ramps[m+2][k]/fdiff[m+1]));}
    }
    // Slaney-style mel is scaled to be approx constant energy per channel;
    float[][] norm_weights = new float[nMels][nFREQS];
    for (int i=0; i<nMels; i++ ) {
        for (int j=0; j<nFREQS; j++ ) {
            norm_weights[i][j] = weights[i][j] * (2.0f / (freqs[i+2] - freqs[i]));}}
    return norm_weights;
}

```

Figura IV.11: JAVA: funciones para el cómputo del espectrograma en escala Mel en Android.

```

private static float[] linSpace(float min, float max, int points) {
    float[] range = new float[points];
    for (int i = 0; i < points; i++) {
        range[i] = min + i * (max - min) / (points-1);
    }
    return range;
}

private static float[][] dot(float[][] x, float[][] y) {
    float[][] z = new float[x.length][y[0].length];
    if(x[0].length==y.length) {
        float sum;
        for (int n = 0; n < x.length; n++) {
            for (int m = 0; m < y[0].length; m++) {
                sum = 0;
                for (int i = 0; i < y.length; i++) {
                    sum += x[n][i] * y[i][m];
                }
                z[n][m] = sum;
            }
        }
        return z;
    }
}

private float[][] power_to_db(float[][] S) {
    float[][] magnitude = new float[S.length][S[0].length];
    // Calculate Magnitude and Reference:
    float ref = S[0][0];
    for (int i=0; i < S.length; i++) {
        for (int j=0; j < S[0].length; j++) {
            magnitude[i][j] = (float)(10.0f * Math.log10(Math.max(1e-10, S[i][j])));
            ref = Math.max(ref,S[i][j]);
        }
    }
    // Convert Log:
    ref = (float) (10.0f * Math.log10(Math.max(1e-10, ref)));
    float magmax = magnitude[0][0] - ref;
    for (int i=0; i < S.length; i++) {
        for (int j=0; j < S[0].length; j++) {
            magnitude[i][j] = magnitude[i][j] - ref;
            magmax = Math.max(magmax, magnitude[i][j]);
        }
    }
    // Get Spectrogram in dB
    for (int i=0; i < S.length; i++) {
        for (int j=0; j < S[0].length; j++) {
            magnitude[i][j] = (float)(Math.max(magnitude[i][j], (magmax - 80.0)));
        }
    }
    return magnitude;
}

private float[][] flipUD(float[][] spec){
    float[][] spectrogramFlipped = new float[nMels][frames];
    for (int i=0; i < nMels; i++) {
        System.arraycopy(spec[(nMels-1-i)], 0, spectrogramFlipped[i], 0, frames);
    }
    return spectrogramFlipped;
}

private float[][] normalize(float[][][] spec){
    float[][] normalizedSpectrogram = new float[nMels][frames];
    for (int j=0 ; j < frames ; j++ ) {
        for (int i = 0; i < nMels; i++) {
            normalizedSpectrogram[i][j] = (spec[i][j]+80)/80;
        }
    }
    return normalizedSpectrogram;
}

```

Figura IV.12: JAVA: funciones traducidas de Numpy para operaciones de matrices en Android.