Universidad Europea de Madrid

Escuela de Arquitectura, Ingeniería y Diseño

Máster en Análisis de Datos Masivos (Big Data)

Trabajo Fin de Máster

Implementación y Evaluación de Modelos de Deep Learning para el Reconocimiento de Emociones Musicales

Autor: Antonio Maqueda Acal

Director: Sergio Iglesias Pérez

Fecha: Marzo, 2026

Madrid, 2026

Resumen

El reconocimiento automático de emociones musicales es una tarea relevante dentro del análisis de audio, especialmente en aplicaciones como los sistemas de recomendación musical, la búsqueda de contenidos y la organización de grandes catálogos. No obstante, se trata de un problema complejo debido tanto a la subjetividad inherente a las emociones como a la elevada variabilidad de la señal musical. Además, en numerosos trabajos previos el análisis se limita a métricas globales, sin profundizar en el comportamiento de los modelos a nivel de emoción individual.

En este Trabajo Fin de Máster se estudia el reconocimiento automático de emociones musicales a partir de audio utilizando el dataset MTG-Jamendo, concretamente el subconjunto *autotagging_moodtheme* (56 etiquetas) y el *split-0* oficial por artista. El proyecto tiene una orientación aplicada y experimental, centrada en la comparación de arquitecturas de Deep Learning bajo una configuración controlada y reproducible. Para ello, se emplean mel-espectrogramas precomputados como representación de entrada y se evalúan tres familias de modelos: una red convolucional (CNN) como línea base, una arquitectura recurrente (CRNN) y un modelo basado en atención (Transformer), manteniendo constante el pipeline y aislando el impacto del modelado temporal.

El trabajo no se limita a la comparación de métricas agregadas, sino que analiza de forma detallada el rendimiento por etiqueta, incluyendo además un análisis estadístico de la significancia de las diferencias observadas. Este enfoque permite identificar patrones de comportamiento, estudiar en qué etiquetas el modelado temporal aporta ventajas o desventajas y comprender por qué determinadas emociones resultan especialmente difíciles de predecir en un escenario multi-etiqueta y desbalanceado.

Como resultado, se obtiene una evaluación comparativa bajo un protocolo controlado, con análisis por etiqueta y validación estadística, que aporta una visión interpretativa y reproducible del comportamiento de distintas arquitecturas en el reconocimiento de emociones musicales.

Palabras clave: reconocimiento de emociones musicales, Deep Learning, clasificación multi-etiqueta, mel-espectrogramas, MTG-Jamendo, mood/theme.

Abstract

Automatic music emotion recognition is a relevant task in audio analysis, with applications in music recommendation, content retrieval, and large-scale catalog organization. However, it remains challenging due to the subjective nature of emotions and the high variability of musical signals. Moreover, many prior works focus on aggregated metrics only, with limited insight into model behavior at the individual emotion level.

This Master's Thesis investigates music emotion recognition from audio using the MTG-Jamendo dataset, specifically the *autotagging_moodtheme* subset (56 mood/theme tags) and the official artist-based *split-0*. The project follows an applied and experimental approach, aiming to compare deep learning architectures under a strictly controlled and reproducible setup. Precomputed mel-spectrograms are used as input representation, and three model families are evaluated: a convolutional neural network (CNN) baseline, a convolutional recurrent neural network (CRNN), and a transformer-based model, while keeping the data pipeline and training protocol consistent to isolate the effect of explicit temporal modeling.

Beyond reporting global performance, the study provides a detailed per-label analysis and includes statistical assessment of observed differences. This enables a more interpretative view of when temporal modeling may help or hurt specific tags, and highlights the impact of class imbalance on evaluation.

Overall, the thesis delivers a controlled comparative evaluation, with per-label analysis and statistical validation, providing an interpretable and reproducible view of how different architectures behave in music mood/theme tagging.

Keywords: music emotion recognition, Deep Learning, multi-label classification, mel-spectrograms, MTG-Jamendo, mood/theme.

Agradecimientos

Este Trabajo Fin de Máster ha sido un reto exigente, pero también una oportunidad que he disfrutado mucho: poder unir algo que siempre me ha acompañado, la música, con un interés que me ha crecido enormemente en los últimos años: el Big Data y el Machine Learning. A medida que avanzaba el proyecto, no solo he aprendido a nivel técnico, sino que también he confirmado que este campo es el camino en el que quiero seguir profundizando. Me voy con más curiosidad, más ambición y la sensación de que esto es solo el principio.

Quiero agradecer, en primer lugar, a mi familia, por su apoyo constante durante todo el proceso. Gracias por vuestra paciencia, por comprender los momentos de estrés y por estar siempre ahí, incluso sin vivir de cerca los tecnicismos de este mundo.

También quiero dar las gracias a mi tutor y a la Universidad, por la orientación y las facilidades ofrecidas durante el desarrollo del trabajo, y a mis compañeros del máster, por el apoyo, el compañerismo y el ambiente de trabajo que han hecho este camino más llevadero.

Por último, me quedo con todo lo aprendido: no solo a nivel técnico, sino también en la forma de plantear problemas, diseñar experimentos y ser riguroso con los resultados.

"All models are wrong, but some are useful."
— George E. P. Box

Tabla Resumen del TFM

Título del TFM	Implementación y Evaluación de Modelos de Deep Learning para el Reconocimiento de Emociones Musicales
Autor	Antonio Maqueda Acal
Director	Sergio Iglesias Pérez
Universidad	Universidad Europea de Madrid
Escuela	Escuela de Arquitectura, Ingeniería y Diseño
Titulación	Máster en Análisis de Datos Masivos (Big Data)
Fecha de entrega	Marzo 2026
Palabras clave	reconocimiento de emociones musicales, Deep Learning, clasificación multietiqueta, mel-espectrogramas, MTG-Jamendo, mood/theme

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Resumen del proyecto

1.1. Contexto y justificación

La forma en que las personas descubren y consumen música ha cambiado sustancialmente en los últimos años. Plataformas de streaming como Spotify o Apple Music utilizan con frecuencia categorías asociadas a estados de ánimo y contextos de uso: listas de reproducción etiquetadas como «música para concentrarse» o «noche tranquila» se han convertido en uno de los principales mecanismos de descubrimiento musical. Detrás de estas clasificaciones existe un problema técnico relevante: asignar automáticamente etiquetas emocionales (*mood/theme*) a pistas musicales a partir únicamente de su señal de audio.

Este problema se enmarca dentro del ámbito de *Music Information Retrieval* (MIR) y presenta una complejidad particular. A diferencia de tareas como la identificación de género o instrumento, donde existen correlatos acústicos relativamente objetivos, la dimensión emocional de la música introduce subjetividad en la anotación, ambigüedad semántica entre etiquetas y una variabilidad que dificulta la generalización de los modelos.

En los últimos años, los avances en Deep Learning aplicado a representaciones espectrales del audio — en particular, mel-espectrogramas procesados con redes neuronales convolucionales y sus variantes — han impulsado mejoras en el rendimiento reportado para tareas de music-tagging. Sin embargo, la comparación entre arquitecturas no siempre es directa: como documentaron Won et al. [2] en su evaluación sistemática de modelos CNN para *music tagging*, una parte significativa de las diferencias de rendimiento reportadas en la literatura puede atribuirse a variaciones en el preprocesado, las particiones de datos o las estrategias de entrenamiento, más que a la arquitectura en sí.

En este contexto, el presente Trabajo Fin de Máster plantea un estudio comparativo controlado entre tres familias de arquitecturas de Deep Learning aplicadas al reconocimiento de emociones musicales, utilizando el subconjunto *mood/theme* del MTG-Jamendo Dataset y un protocolo experimental diseñado para aislar el impacto del mecanismo de agregación temporal, manteniendo constantes el resto de factores.

1.2. Planteamiento del problema

El reconocimiento de emociones musicales se formula como un problema de clasificación multi-etiqueta: una misma canción puede estar asociada simultáneamente a varias emociones (por ejemplo, *calm*, *romantic* y *melancholic* a la vez). La literatura ha propuesto distintos enfoques para abordar esta tarea, incluyendo modelos basados en redes convolucionales (CNN), combinaciones de CNN con redes recurrentes (CRNN) y, más recientemente, modelos basados en mecanismos de atención de tipo Transformer.

Sin embargo, dos problemas limitan el estado actual del conocimiento:

1. Las comparaciones entre arquitecturas suelen estar condicionadas por cambios simultáneos en múltiples factores experimentales (representación de entrada, duración de los fragmentos, estrategia de entrenamiento, uso de pre-entrenamiento o aumentación), lo que dificulta atribuir las diferencias de rendimiento al modelado temporal en sí.

2. El análisis se limita habitualmente a métricas agregadas (macro ROC-AUC, macro PR-AUC), sin profundizar en el comportamiento de los modelos a nivel de emoción individual ni validar estadísticamente la estabilidad de las diferencias observadas. En tareas multi-etiqueta desbalanceadas, esta práctica puede ocultar que dos arquitecturas con rendimiento global similar se comporten de manera muy distinta por etiqueta.

Este proyecto plantea un estudio comparativo que aborda ambas limitaciones: un protocolo experimental controlado que aísla el mecanismo de agregación temporal como única variable, complementado con un análisis por etiqueta emocional y con procedimientos de validación estadística que permiten interpretar las diferencias con mayor rigor.

1.3. Objetivos del proyecto

El objetivo de este proyecto es analizar y comparar el comportamiento de distintas arquitecturas de Deep Learning aplicadas al reconocimiento de emociones musicales a partir de audio, bajo una configuración experimental controlada y reproducible.

Para ello, se evalúan tres familias de modelos — redes neuronales convolucionales (CNN), redes convolucionales recurrentes (CRNN) y arquitecturas CNN-Transformer — que comparten un mismo *backbone* convolucional y varían principalmente en cómo agregan la información temporal. Se utilizan mel-espectrogramas como representación de entrada y se mantienen constantes el dataset, las particiones y el protocolo de entrenamiento.

El análisis se centra tanto en el rendimiento global de los modelos como en su comportamiento por etiqueta emocional, incorporando además procedimientos de validación estadística y un estudio del efecto del umbral de decisión en métricas dependientes de umbral. El objetivo último es comprender mejor las diferencias entre arquitecturas, las dificultades asociadas a determinadas etiquetas y el papel real del modelado temporal en esta tarea.

1.4. Resultados obtenidos

Como resultado del proyecto se obtiene una implementación funcional y reproducible de tres modelos de Deep Learning evaluados sobre el subconjunto *autotagging_moodtheme* del MTG-Jamendo Dataset (56 etiquetas), utilizando el *split-0* oficial por artista para minimizar la ausencia de *data leakage* entre conjuntos.

A partir de las evaluaciones realizadas, se dispone de una comparación homogénea entre las tres arquitecturas bajo un protocolo experimental común. Asimismo, se obtiene un análisis detallado del rendimiento por etiqueta emocional, que permite identificar diferencias entre emociones, detectar aquellas especialmente difíciles de predecir y discutir posibles causas relacionadas con la naturaleza de la señal musical, el desbalance de etiquetas y el tipo de modelado empleado. Adicionalmente, se incorporan análisis complementarios — incluidos el ajuste de umbrales de decisión y la estimación de incertidumbre estadística — que refuerzan la interpretabilidad y la robustez de las conclusiones.

En conjunto, el proyecto permite extraer conclusiones interpretables sobre el papel del modelado temporal en el reconocimiento de emociones musicales y sobre qué factores — asociados a la arquitectura o a las características de las propias etiquetas — condicionan el rendimiento de los modelos.

1.5. Estructura de la memoria

Este documento comienza con una revisión del estado del arte en el reconocimiento automático de emociones musicales, donde se analizan los principales enfoques y limitaciones existentes. A continuación, se definen los objetivos del proyecto y se describe el desarrollo del trabajo, incluyendo la metodología experimental y los modelos evaluados. Posteriormente, se presentan los resultados obtenidos y se realiza un análisis detallado de los mismos, con especial atención al comportamiento por etiqueta emocional. Finalmente, se recogen las conclusiones del trabajo y se proponen posibles líneas de trabajo futuro.

Capítulo 2

Antecedentes / Estado del arte

En este apartado se revisan los conceptos y enfoques más relevantes del reconocimiento automático de emociones musicales a partir de audio. El objetivo es proporcionar el contexto necesario para entender cómo se representa el audio, cómo se formula el problema como clasificación multi-etiqueta y qué familias de modelos se han utilizado habitualmente.

Esta revisión permite justificar las decisiones metodológicas del proyecto y el enfoque comparativo adoptado en los capítulos posteriores.

2.1. Reconocimiento de emociones musicales en MIR

El reconocimiento automático de emociones musicales (*Music Emotion Recognition*, MER) es una tarea consolidada dentro del ámbito de *Music Information Retrieval* (MIR). Tal como recogen Yang y Chen [3] en su revisión del campo, el objetivo de MER es identificar los estados afectivos o contextuales que transmite una pieza musical mediante el análisis de su señal de audio. A diferencia de tareas más «objetivas» como la identificación de género o instrumento, la dimensión emocional introduce una complejidad adicional: el significado de una etiqueta puede depender de la percepción del oyente, del contexto cultural y del uso previsto de la música.

Desde el punto de vista conceptual, la literatura en MIR distingue habitualmente entre varios términos relacionados pero con matices distintos. En el modelo dimensional clásico de Russell [4], las emociones se representan en un espacio continuo definido por ejes de valencia y activación (*arousal*), mientras que Thayer [5] propuso una categorización biopsicológica que diferencia entre energía y tensión. Eerola y Vuoskoski [6] compararon empíricamente los modelos dimensional y categórico en el contexto musical, concluyendo que ambos capturan aspectos complementarios de la percepción emocional. En la práctica aplicada a MIR, se emplean habitualmente las categorías de *emotion*, *mood* y *theme*: las *emotions* suelen asociarse a estados afectivos relativamente directos (por ejemplo, *happy* o *sad*).

Los *moods* describen estados más difusos o sostenidos en el tiempo (por ejemplo, *calm* o *melancholic*). Por su parte, los *themes* hacen referencia a contextos o usos habituales de una pieza musical (por ejemplo, *film*, *epic* o *travel*) y, en muchos casos, no tienen un correlato emocional directo ni un patrón acústico inequívoco [1, 7]. Esta heterogeneidad semántica es especialmente relevante en el subconjunto *mood/theme*, donde coexisten etiquetas afectivas y etiquetas de contexto. Delbouys et al. [8] ya señalaron la dificultad intrínseca de predecir dimensiones afectivas como la valencia a partir únicamente de audio, lo que refuerza la idea de que ciertos aspectos emocionales pueden ser inherentemente difíciles de capturar con señales acústicas.

Desde una perspectiva computacional, diversos trabajos en MIR han formulado el reconocimiento de emociones musicales como un problema de clasificación multi-etiqueta [9], ya que una misma canción puede estar asociada simultáneamente a varias etiquetas. Por ejemplo, una pista puede describirse como *calm*, *romantic* y *melancholic* al mismo tiempo. Esto implica que las etiquetas no son mutuamente excluyentes y que pueden existir relaciones de dependencia o correlación entre ellas (por ejemplo, ciertas combinaciones son más probables que otras). En consecuencia, el objetivo del modelo no es elegir una única clase «correcta», sino estimar, para cada etiqueta, la probabilidad de que esté presente en la canción.

A esta complejidad se suma un desafío práctico bien documentado: el desbalance entre etiquetas. En la mayoría de *datasets* de *auto-tagging* musical, algunas etiquetas son muy frecuentes mientras que otras aparecen en una proporción reducida de ejemplos [10]. Este fenómeno afecta tanto al entrenamiento como a la evaluación, ya que pequeñas variaciones en las predicciones de etiquetas raras pueden provocar cambios grandes en métricas sensibles a la prevalencia.

En el caso del *mood/theme*, este desbalance se combina con la subjetividad y la ambigüedad semántica, lo que explica que ciertos *moods* y *themes* continúen siendo difíciles de predecir de forma fiable incluso con modelos avanzados.

En los últimos años, este problema se ha abordado principalmente mediante técnicas de *Deep Learning* aplicadas a representaciones espectrales del audio. Como documentaron Choi et al. [11], estos enfoques permiten aprender automáticamente patrones complejos a partir de la señal tiempo-frecuencia, incorporando distintas formas de agregación temporal (implícitas o explícitas) para capturar aspectos musicales relevantes para la percepción emocional. No obstante, la evidencia reportada en la literatura sugiere que el rendimiento varía de forma notable entre etiquetas y que una parte importante de la dificultad del problema proviene de la propia naturaleza de las etiquetas y del marco de anotación, además de las limitaciones de las arquitecturas empleadas [2].

2.2. De audio a mel-espectrograma

Para aplicar modelos de *Deep Learning* al análisis musical es necesario transformar la señal de audio a una representación adecuada para el aprendizaje automático. En MIR, la representación más extendida es el mel-espectrograma, una representación bidimensional tiempo-frecuencia que captura información perceptualmente relevante del contenido sonoro [12]. El objetivo de esta transformación es separar de forma explícita la información temporal y frecuencial, facilitando que los modelos aprendan patrones acústicos relevantes para la tarea de reconocimiento de emociones.

2.2.1. De señal temporal a representación tiempo-frecuencia (STFT)

Una pista musical en su forma original es una señal unidimensional $x(t)$ que describe la variación de la presión sonora a lo largo del tiempo. Sin embargo, esta representación no separa de forma directa componentes clave para la percepción musical, como el timbre, la armonía o la energía por bandas de frecuencia.

Por ello, se emplea la *Short-Time Fourier Transform* (STFT), una herramienta fundamental en procesamiento de señal de audio [12, 13]. La STFT analiza el audio en ventanas sucesivas de longitud fija (`n_fft`), desplazadas por un paso (`hop_length`), y aplica la transformada de Fourier a cada ventana. El resultado es un espectrograma complejo $S(f,t) \in \mathbb{C}$ donde:

- un eje representa el tiempo (ventanas sucesivas),
- el otro eje representa la frecuencia (bins de Fourier),
- cada celda indica la magnitud de energía en esa frecuencia y ese instante temporal.

A cada segmento se le aplica previamente una función de ventana —habitualmente la ventana de Hann— para reducir el *spectral leakage*: sin ella, la discontinuidad en los bordes del segmento introduciría artefactos espectrales que distorsionarían la representación [12]. De la salida compleja se retiene únicamente la magnitud $|S(f,t)|$ —o el espectro de potencia $|S(f,t)|^2$—, descartando la información de fase, que no resulta relevante para la percepción tímbrica ni para las tareas de clasificación de contenido musical.

La elección del tamaño de ventana implica un compromiso entre resolución temporal y frecuencial: ventanas más largas proporcionan mayor resolución frecuencial pero menor resolución temporal, y viceversa. Müller [12] describe en detalle esta relación, conocida como principio de incertidumbre tiempo-frecuencia.

En la práctica, valores típicos en MIR son `n_fft` = 2048 y `hop_length` = 512 muestras a una frecuencia de muestreo de 22.050 Hz, lo que proporciona un equilibrio razonable para señales musicales. La figura 2.1 ilustra este pipeline sobre una señal sintética: en (b) se aprecian cuatro bandas de energía horizontales correspondientes a los cuatro componentes armónicos (220, 440, 660 y 880 Hz); en (c), esas mismas bandas aparecen comprimidas hacia el eje inferior, reflejo de que la escala mel concentra la resolución en frecuencias graves.

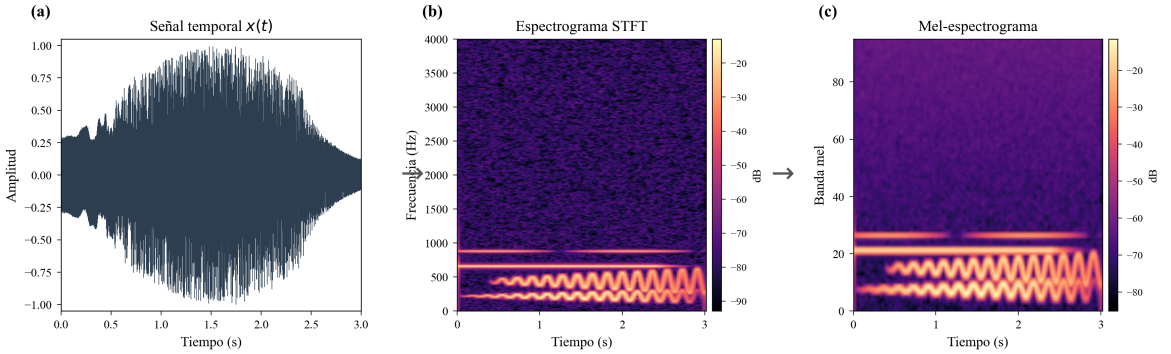

Figura 2.1: Pipeline de representación del audio sobre una señal sintética (22.050 Hz, `n_fft` = 2048, `hop_length` = 512). **(a)** Señal temporal $x(t)$: variación de amplitud a lo largo de 3 segundos. **(b)** Espectrograma STFT: cada columna es el espectro de una ventana temporal; el color indica energía en dB (amarillo claro = alta energía, negro = baja energía); las cuatro líneas horizontales corresponden a los armónicos a 220, 440, 660 y 880 Hz. **(c)** Mel-espectrograma (96 bandas mel, escala logarítmica): los mismos armónicos aparecen concentrados en las bandas bajas, ya que la escala mel asigna mayor resolución a frecuencias graves.

2.2.2. Mel-espectrograma: motivación e interpretación

El mel-espectrograma es una variante perceptualmente motivada del espectrograma, en el que el eje de frecuencias se proyecta a la escala mel. Esta escala perceptual, propuesta originalmente por Stevens et al. [14], aproxima cómo el oído humano discrimina diferencialmente las frecuencias a lo largo del rango audible. La proyección agrupa frecuencias de forma no lineal mediante un banco de filtros triangulares solapantes (*mel filterbank*), aplicado sobre el espectro de potencia $|S(f,t)|^2$. El solapamiento entre filtros adyacentes garantiza una cobertura espectral continua y sin huecos, concentrando mayor resolución en rangos perceptualmente relevantes (típicamente por debajo de 1 kHz) y comprimiendo las frecuencias altas donde la discriminación auditiva es menor [12].

Formalmente, la conversión de frecuencia en hercios (f) a la escala mel se define como:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.1}$$

El resultado de aplicar el banco de filtros mel al espectrograma STFT es una matriz $\mathbf{M} \in \mathbb{R}^{F \times T}$ donde F son bandas mel y T *frames* temporales. Cada valor $M_{f,t}$ representa la energía en una banda de frecuencia f en el instante t. Es habitual aplicar una compresión logarítmica a los valores de energía ($\log(1 + M_{f,t})$ o $10 \cdot \log_{10}(M_{f,t})$), lo que reduce el rango dinámico y aproxima la percepción logarítmica de la intensidad sonora del oído humano [12].

Esta estructura bidimensional puede interpretarse como una «imagen tiempo-frecuencia», lo que justifica el uso extendido de arquitecturas convolucionales en tareas de *music auto-tagging*. La efectividad de esta representación como entrada para CNN fue demostrada por Choi et al. [11] en MagnaTagATune y confirmada por diversos trabajos posteriores [15, 16]. Hershey et al. [16] mostraron además que esta misma aproximación (CNN sobre mel-espectrogramas) escala eficazmente a tareas de clasificación de audio a gran escala como AudioSet.

El número de bandas mel (F) es un hiperparámetro del preprocesado. Valores habituales en la literatura de MIR oscilan entre 64 y 128 bandas. En este proyecto se utilizan $F = 96$ bandas, un valor consistente con el empleado en el *baseline* de MTG-Jamendo [7].

2.2.3. Decisión práctica: mel-espectrogramas precomputados y ventana fija

En este proyecto se trabaja directamente con mel-espectrogramas precomputados almacenados en archivos `.npy`, con una resolución de $F = 96$ bandas mel y una dimensión temporal T variable según la duración de cada pista. Esta estrategia, ampliamente utilizada en literatura de MIR [7], evita recalcular el preprocesado del audio y garantiza que todos los modelos comparados reciben exactamente la misma representación de entrada, mejorando la reproducibilidad y la comparabilidad experimental.

Adicionalmente, para asegurar una entrada de dimensión constante, cada mel-espectrograma se ajusta a una ventana temporal fija de $T = 1366$ *frames* mediante recorte o relleno centrado (*center crop/pad*). De este modo, todas las arquitecturas (CNN, CRNN y Transformer) operan sobre entradas homogéneas de dimensión $(96, 1366)$, manteniendo la comparabilidad experimental. Esta estandarización de entrada es fundamental para atribuir diferencias de rendimiento exclusivamente a la arquitectura y no a variaciones en el preprocesado.

2.3. Salida multi-etiqueta: *multi-hot*, *sigmoid* y función de pérdida

Una vez definida la representación de entrada (mel-espectrograma), el siguiente aspecto clave es cómo se representa la salida del modelo y cómo se entrena en un escenario donde una canción puede tener varias etiquetas emocionales simultáneamente.

2.3.1. Clasificación multi-clase *vs* multi-etiqueta

En un problema multi-clase, cada ejemplo pertenece a una única clase (por ejemplo, elegir un único género). En ese caso, las clases son mutuamente excluyentes y la salida típica del modelo se interpreta como una distribución de probabilidad que suma 1.

En cambio, en el reconocimiento *mood/theme* se trata de un problema multi-etiqueta (*multi-label*) [9]: una misma canción puede ser *calm* y *romantic* a la vez. Por tanto, las etiquetas no son excluyentes, y el modelo debe estimar, de forma independiente, si cada etiqueta está presente o no. Este cambio conceptual tiene implicaciones profundas en la formulación matemática, la función de pérdida empleada y la interpretación de las métricas de evaluación.

2.3.2. Vector *multi-hot* y probabilidades por etiqueta

Para representar este tipo de salida se utiliza un vector binario de dimensión L (en este proyecto, $L = 56$). Cada componente del vector indica la presencia (1) o ausencia (0) de una etiqueta:

$$\mathbf{y} \in \{0, 1\}^{56} \tag{2.2}$$

A esta codificación se le suele denominar *multi-hot* (o *multi-hot vector*), ya que puede contener varios unos simultáneamente [9].

El modelo produce como salida un vector de valores reales (*logits*), uno por etiqueta. Estos *logits* se transforman en probabilidades aplicando una función *sigmoid* componente a componente:

$$p_j = \sigma(z_j) = \frac{1}{1 + e^{-z_j}} \tag{2.3}$$

donde z_j es el *logit* de la etiqueta j y p_j su probabilidad estimada. A diferencia del caso multi-clase, aquí no se usa *softmax*, porque no se quiere forzar que solo una etiqueta sea la correcta ni que las

probabilidades sumen 1 [17].

2.3.3. Entrenamiento: `BCEWithLogitsLoss` y decisión por umbral

Dado que cada etiqueta se comporta como un problema binario («está» / «no está»), el entrenamiento se plantea como la suma de L pérdidas binarias independientes [17]. La función utilizada habitualmente es la entropía cruzada binaria (*Binary Cross-Entropy*), que en PyTorch se implementa de forma numéricamente estable como `BCEWithLogitsLoss`, combinando internamente el *sigmoid* y la pérdida en una sola operación logarítmica.

En fase de inferencia, para convertir probabilidades en predicciones binarias se aplica un umbral t: si $p_j \geq t$, se predice que la etiqueta j está presente. Un valor habitual por defecto es $t = 0{,}5$, pero en problemas desbalanceados este umbral puede no ser apropiado, ya que una etiqueta rara puede requerir umbrales más bajos para recuperar suficientes positivos. Este hecho ha sido ampliamente documentado en literatura de clasificación desbalanceada.

Por este motivo, es común complementar las métricas de *ranking* (como ROC-AUC o PR-AUC) con análisis basados en umbrales ajustados, especialmente cuando se desea interpretar F1 o precisión/recobrado en etiquetas poco frecuentes. El *baseline* de MediaEval 2019 ya aplica este procedimiento, calculando umbrales por etiqueta que maximizan el *F-score* macro sobre el conjunto de validación [1].

2.4. Modelos de *Deep Learning* para *auto-tagging* emocional

Una vez definidas la representación de entrada (mel-espectrograma) y la salida multi-etiqueta (vector *multi-hot*), la literatura ha propuesto distintas familias de arquitecturas para aprender la relación entre ambas. En *music auto-tagging* emocional, estas arquitecturas pueden entenderse como distintas formas de extraer patrones tiempo-frecuencia y agregar información temporal para producir una predicción por etiqueta.

2.4.1. CNN como *baseline*: patrones locales y agregación global

Las redes neuronales convolucionales (CNN), cuya formulación moderna fue establecida por LeCun et al. [18], constituyen el enfoque de referencia en *auto-tagging* musical desde principios de la década de 2010. Al operar sobre mel-espectrogramas, las CNN aprenden filtros locales que detectan patrones en regiones pequeñas del dominio tiempo-frecuencia (por ejemplo, concentraciones de energía en ciertas bandas, texturas tímbricas o estructuras repetitivas).

Mediante capas de *pooling*, estas representaciones se van comprimiendo y agregando, permitiendo que el modelo capture información cada vez más abstracta.

Choi et al. [11] demostraron que una arquitectura CNN de 4 capas con mel-espectrogramas como entrada alcanza rendimiento competitivo en MagnaTagATune, y que arquitecturas más profundas mejoran los resultados cuando se dispone de más datos de entrenamiento. Posteriormente, el propio *benchmark* MTG-Jamendo adoptó como *baseline* un modelo CNN de 5 capas con filtros 3×3 siguiendo esta línea [7]. Lee et al. [15] exploraron un enfoque alternativo operando directamente sobre *waveforms* a nivel de muestra, confirmando que los filtros aprendidos capturan sensibilidad frecuencial similar a la del mel-espectrograma.

En este tipo de tareas, las CNN suelen emplearse como *baseline* porque aportan un buen equilibrio entre capacidad de aprendizaje y coste computacional. Sin embargo, una limitación conocida reportada en la literatura es que, aunque la dimensión temporal está presente en la entrada, la red no incorpora necesariamente un mecanismo explícito para modelar dependencias temporales de largo alcance: gran parte de la secuencia se resume mediante operaciones de *pooling* y agregación global.

2.4.2. CRNN: CNN como extractor + RNN para modelado temporal explícito

Para capturar de forma más directa la evolución temporal, se han propuesto arquitecturas híbridas conocidas como CRNN (*Convolutional Recurrent Neural Networks*). Estas arquitecturas combinan la capacidad de la CNN para extraer características locales con redes recurrentes —en particular *Long Short-Term Memory* (LSTM) [19]— capaces de modelar dependencias secuenciales. En el trabajo seminal de Choi et al. [20], se propuso una arquitectura que utiliza la CNN como extractor de características locales (típicamente a escala de *frames* cortos), reorganiza su salida como una secuencia temporal y la procesa con una capa LSTM. La arquitectura LSTM, propuesta por Hochreiter y Schmidhuber [19], resuelve el problema del desvanecimiento del gradiente en redes recurrentes convencionales mediante puertas de entrada, olvido y salida que regulan el flujo de información a través de la secuencia. Los autores demostraron que esta combinación CNN+LSTM alcanza rendimiento competitivo con respecto a modelos CNN de mayor tamaño, manteniendo un número reducido de parámetros y un tiempo de entrenamiento favorable.

La intuición detrás de este enfoque es que la parte recurrente puede integrar información a lo largo del tiempo y capturar dependencias que no se describen bien con patrones locales aislados: por ejemplo, transiciones progresivas de intensidad, cambios de densidad sonora o estructuras musicales que requieren contexto temporal. En consecuencia, las CRNN se han empleado ampliamente en tareas donde la dinámica temporal es relevante. Won et al. [2], en su evaluación sistemática de modelos CNN para *music tagging*, observaron que las diferencias de rendimiento entre arquitecturas tienden a ser menores de lo esperado cuando se controlan adecuadamente los factores experimentales, una observación que motivó en parte el planteamiento comparativo de este trabajo.

2.4.3. Modelos con atención y Transformers: dependencias largas sin recurrencia

Más recientemente, se han explorado modelos basados en mecanismos de atención y arquitecturas de tipo Transformer. Como estableció el trabajo fundamental de Vaswani et al. [21], estos enfoques permiten modelar dependencias temporales sin procesar la secuencia de forma estrictamente secuencial (como en un RNN), sino estableciendo relaciones directas entre distintos instantes temporales mediante *self-attention*.

Won et al. [22] propusieron una de las primeras aplicaciones de *self-attention* a *music tagging*, combinando capas convolucionales superficiales con *Transformer encoders* apilados. Los autores demostraron que este enfoque alcanza rendimiento competitivo en MagnaTagATune y Million Song Dataset, con la ventaja adicional de producir mapas de atención interpretables que visualizan qué regiones tiempo-frecuencia son relevantes para cada predicción.

En el ámbito más amplio de la clasificación de audio, Gong et al. [23] propusieron el *Audio Spectrogram Transformer* (AST), el primer modelo puramente basado en atención (sin convoluciones) para clasificación de audio, alcanzando resultados destacados en AudioSet. Chen et al. [24] introdujeron HTS-AT, un Transformer jerárquico que reduce sustancialmente el coste computacional (35 % de parámetros y 15 % del tiempo de entrenamiento respecto a enfoques Transformer previos) manteniendo rendimiento competitivo. Más recientemente, modelos de pre-entrenamiento auto-supervisado a gran escala como MERT [25] han demostrado la capacidad de los Transformers para aprender representaciones musicales generalizables a múltiples tareas *downstream*.

En términos intuitivos, la atención permite que el modelo «decida» qué partes de la secuencia son relevantes para predecir una etiqueta concreta, pudiendo relacionar eventos separados en el tiempo. Esto resulta especialmente atractivo cuando se busca capturar dependencias de largo alcance o cuando se sospecha que la emoción percibida depende de momentos concretos de la pieza. No obstante, como han señalado trabajos recientes, estos modelos suelen ser más costosos computacionalmente y su rendimiento está fuertemente condicionado por decisiones de entrenamiento como la tasa de aprendizaje, el tamaño de *batch* y el uso de técnicas adicionales como preentrenamiento o augmentación [2, 23].

2.5. Métricas en la literatura: ROC-AUC y PR-AUC

La evaluación de modelos en clasificación multi-etiqueta requiere métricas que reflejen adecuadamente el rendimiento en presencia de múltiples etiquetas y, especialmente, de desbalance entre clases. En *music auto-tagging* emocional es habitual que algunas etiquetas sean frecuentes y otras aparezcan en una proporción muy reducida de canciones. En este contexto, métricas basadas únicamente en exactitud o en un umbral fijo pueden resultar poco informativas.

2.5.1. Evaluación por etiqueta y promedios agregados

En problemas multi-etiqueta, el rendimiento se calcula típicamente de forma independiente para cada etiqueta (tratando cada una como un problema binario). Posteriormente, se agregan los resultados mediante promedios. El promedio más utilizado en este tipo de *benchmarks* es el *macro-average*, que calcula la media aritmética del rendimiento de cada etiqueta, asignando el mismo peso a etiquetas frecuentes y raras. Esta elección permite analizar la tarea desde una perspectiva «por concepto», evitando que unas pocas etiquetas dominantes oculten el comportamiento en el resto.

2.5.2. ROC-AUC: métrica estándar para capacidad de *ranking*

La métrica ROC-AUC (*Area Under the Receiver Operating Characteristic Curve*) mide la capacidad del modelo para ordenar correctamente ejemplos positivos por encima de negativos para una etiqueta dada [26]. Un valor de 0,5 equivale al azar, mientras que valores cercanos a 1,0 indican separación casi perfecta. Como describe Fawcett [26], la curva ROC se construye variando el umbral de decisión y representando la tasa de verdaderos positivos frente a la tasa de falsos positivos; el área bajo esta curva resume la capacidad discriminativa del clasificador en un único escalar.

Una ventaja importante de ROC-AUC es que es independiente del umbral, lo que la hace adecuada cuando se quiere evaluar la calidad del *ranking* de probabilidades sin fijar una decisión binaria. Por este motivo, ROC-AUC se ha consolidado como métrica principal en distintos trabajos de *auto-tagging* y en *benchmarks* como MTG-Jamendo y MediaEval [2, 7].

2.5.3. PR-AUC / *Average Precision*: sensibilidad al desbalance

La métrica basada en precisión-*recall* (PR) y su área (PR-AUC, frecuentemente reportada como *Average Precision*, AP) es especialmente relevante cuando existe un fuerte desbalance entre positivos y negativos. A diferencia de ROC-AUC, que considera tanto verdaderos positivos como verdaderos negativos, PR-AUC se centra en la capacidad del modelo para recuperar correctamente los positivos manteniendo una precisión razonable.

Davis y Goadrich [27] demostraron formalmente que un algoritmo que domina en el espacio ROC también domina en el espacio PR, pero que la relación inversa no se cumple necesariamente. Esto implica que PR-AUC puede revelar debilidades que ROC-AUC no captura, particularmente cuando los negativos son mucho más numerosos que los positivos. Por este motivo, el propio *paper* del MTG-Jamendo justifica reportar PR-AUC junto a ROC-AUC citando este resultado [7].

En etiquetas raras, PR-AUC tiende a ser más exigente y a reflejar de forma más directa la dificultad práctica del problema. Por ello, es habitual reportar conjuntamente ROC-AUC y PR-AUC/AP: mientras la primera informa sobre separación global, la segunda aporta una visión más realista del rendimiento cuando los positivos son escasos.

2.6. Contexto y justificación del *benchmark* adoptado

El análisis del estado del arte muestra que el reconocimiento automático de emociones musicales ha avanzado gracias a arquitecturas de *Deep Learning* y a la existencia de *benchmarks* estandarizados. Sin embargo, como documentaron Won et al. [2], en esta tarea es especialmente fácil «contaminar» la comparación entre modelos si se modifican simultáneamente variables como la representación de entrada, las particiones, el preprocesado o el uso de preentrenamiento. En su evaluación sistemática sobre MagnaTagATune [28], Million Song Dataset [29] y MTG-Jamendo, los autores observaron que las diferencias de rendimiento entre arquitecturas CNN se reducen sustancialmente cuando se homogeneizan las condiciones experimentales, concluyendo que gran parte de la variabilidad reportada en la literatura se debe a factores ajenos a la arquitectura en sí.

Por ello, este TFM se apoya en el *benchmark* MTG-Jamendo y adopta un protocolo experimental controlado orientado a aislar el impacto del modelado temporal (ausente *vs* recurrente *vs* atención) manteniendo constantes el resto de factores.

2.6.1. MTG-Jamendo como *benchmark*

El MTG-Jamendo Dataset se propuso explícitamente como un *dataset* abierto para *music auto-tagging* que solucionase limitaciones frecuentes en *datasets* previos (por ejemplo, uso de segmentos cortos, calidad de audio inconsistente o falta de particiones estandarizadas) [7].

En el artículo original se indica que el *dataset* contiene 55.701 pistas completas ($\geq$ 30 s), con audio MP3 de alta calidad y un volumen total de 3.777 horas, proporcionando además metadatos con identificadores de pista, artista y álbum [7]. Esta escala y consistencia lo convierten en una referencia adecuada para evaluar modelos con capacidad de generalización.

Para el presente trabajo se utiliza el subconjunto `autotagging_moodtheme`, adoptado como referencia en el reto MediaEval 2019. En el *overview* de MediaEval se especifica que este subconjunto incluye 18.486 pistas y 56 etiquetas distintas, y que cada pista tiene al menos una etiqueta, pudiendo tener varias simultáneamente (naturaleza multi-etiqueta) [1].

Nota sobre el número de etiquetas. El *paper* original del *dataset* reporta 59 etiquetas *mood/theme* en el *dataset* base. Sin embargo, al construir las particiones oficiales (*splits*), se filtran aquellas etiquetas que no disponen de suficientes ejemplos para garantizar la misma lista de *tags* en todos los *splits*, resultando finalmente en 56 etiquetas disponibles en las particiones, que son las que se utilizan en este trabajo.

2.6.2. *Splits* oficiales y prevención de *data leakage*

Una contribución metodológica clave de MTG-Jamendo es la definición de particiones (*splits*) que evitan fugas de información por artista o álbum. En el *paper* del *dataset* se explica que los *splits* se generan imponiendo que ni validación ni test contengan pistas de los mismos artistas/álbumes que el entrenamiento [7].

Este criterio mitiga el denominado *artist effect/album effect*: si un artista aparece en entrenamiento y también en test, un modelo puede «reconocer al artista» (producción, timbre, mezcla, estilo) en lugar de aprender patrones acústicos generalizables asociados a las etiquetas. Por ello, usar *splits* por artista es esencial para que el rendimiento refleje generalización real.

En el marco de MediaEval, además, se enfatiza el uso correcto de cada partición: validación para *early stopping* y ajuste de hiperparámetros, y test únicamente para estimar rendimiento final [1].

En este proyecto se utiliza exclusivamente el *split-0* del subconjunto *mood/theme*. La distribución resultante es asimétrica respecto a los ratios generales del *dataset* (que Bogdanov et al. [7] reportan como aproximadamente 60/20/20 para el conjunto completo), ya que el filtrado por etiquetas y artistas en el subconjunto *mood/theme* produce proporciones distintas.

2.6.3. *Baselines* y trabajos previos en MTG-Jamendo

El *benchmark* aporta *baselines* cuantificados que sirven como referencia reproducible.

***Baseline* en el *paper* del *dataset* MTG-Jamendo.** En el artículo de MTG-Jamendo se reporta un *baseline* basado en el modelo propuesto por Choi et al. [11]: una CNN de 5 capas con filtros 3×3, entrenada durante 100 épocas, usando un segmento centrado de 29,1 s por pista y reportando métricas macro por etiqueta.

Los autores reportan resultados para distintos subconjuntos; para *Mood/Theme* (18.486 pistas; 1.533 artistas; 59 *tags* en el *dataset* base), el *paper* reporta ROC-AUC = 67,19 y PR-AUC = 8,19 (en la tabla se presentan en escala 0–100) [7, Table 2].

El propio *paper* justifica reportar PR-AUC junto a ROC-AUC porque ROC-AUC puede ser optimista con datos desbalanceados, citando a Davis y Goadrich [27].

***Baselines* oficiales en MediaEval 2019 (VGG-ish).** En el *overview* de la tarea MediaEval 2019 se define un *baseline* principal basado en una arquitectura tipo VGG-ish [30] (cinco capas convolucionales 2D seguidas de una capa densa). Se indica que entrenan durante 1000 épocas, seleccionan el mejor modelo con validación y calculan umbrales por etiqueta maximizando *F-score* macro. Los resultados del *baseline* en test se resumen en la cuadro 2.1; además, el *overview* reporta precisión y recobrado macro y micro [1].

Tabla 2.1: Resultados del *baseline* VGG-ish en MediaEval 2019 [1].

Métrica	**Valor**
ROC-AUC	0,725
PR-AUC	0,107
F-score macro	0,165
F-score micro	0,177

Estos *baselines* refuerzan dos ideas centrales para el TFM:

1. El *benchmark* está diseñado para evaluar modelos con métricas multi-etiqueta estándar (macro ROC-AUC y PR-AUC/AP como métricas de *ranking*, y *F-scores* como métricas dependientes de umbral).

2. PR-AUC/AP es especialmente relevante porque el desbalance de etiquetas hace que ROC-AUC pueda sobreestimar rendimiento cuando los negativos dominan [7, 27].

Es importante señalar que el *baseline* VGG-ish de MediaEval se entrena con un protocolo sustancialmente más intensivo que el adoptado en este TFM: 1000 épocas frente a 100, umbrales optimizados por etiqueta para las métricas de *F-score*, y una configuración orientada a maximizar rendimiento absoluto. Dado que el objetivo de este trabajo no es competir en rendimiento sino aislar el efecto del modelado temporal, se opta deliberadamente por un protocolo más parco, lo que implica que los rendimientos absolutos serán inferiores a los de estos *baselines*. Esta decisión se justifica en detalle en el capítulo de metodología.

2.7. Planteamiento del problema

A partir del estado del arte revisado, se identifica un punto crítico: aunque existen múltiples arquitecturas aplicadas al reconocimiento de emociones musicales (*mood/theme*), no siempre es posible atribuir las diferencias de rendimiento observadas a la arquitectura en sí.

En muchos trabajos, la comparación entre modelos se ve condicionada por cambios simultáneos en factores como el preprocesado, la duración de los fragmentos analizados, la estrategia de entrenamiento, el uso de *data augmentation* o el preentrenamiento. Won et al. [2] documentaron formalmente este problema, mostrando que bajo condiciones experimentales controladas las diferencias entre arquitecturas CNN se reducen de forma notable. Esta heterogeneidad dificulta aislar el efecto real del modelado temporal y limita la comparabilidad entre enfoques.

En particular, dos cuestiones siguen abiertas desde una perspectiva experimental:

1. Si incorporar mecanismos explícitos de modelado temporal (recurrentes o basados en atención) aporta una mejora consistente frente a una CNN que agrega la información de forma más estática.

2. Si dicho beneficio —cuando existe— se distribuye de forma homogénea entre etiquetas o, por el contrario, se concentra en un subconjunto de *moods/themes* con propiedades específicas (por ejemplo, etiquetas asociadas a estructura musical global o a transiciones temporales).

Además, el enfoque habitual basado únicamente en métricas agregadas puede ocultar aspectos relevantes del comportamiento del modelo. En tareas multi-etiqueta desbalanceadas, dos arquitecturas pueden obtener resultados globales similares y, sin embargo, comportarse de manera distinta por etiqueta: algunas emociones pueden beneficiarse del contexto temporal mientras que otras pueden depender principalmente de patrones espectrales locales o estar limitadas por la ambigüedad de la etiqueta y la escasez de ejemplos. Por tanto, un análisis riguroso requiere complementar las métricas globales con una evaluación por etiqueta y con herramientas que permitan interpretar la estabilidad de las diferencias observadas.

En este contexto, este Trabajo Fin de Máster plantea un estudio experimental controlado con el objetivo de cubrir ese vacío: evaluar de forma sistemática el impacto del modelado temporal en el reconocimiento de emociones musicales, utilizando un *benchmark* estandarizado y manteniendo constantes los elementos esenciales del *pipeline*.

Para ello, se comparan tres familias representativas bajo condiciones homogéneas:

- una CNN como línea base (sin modelado temporal explícito),
- una CRNN (CNN + red recurrente) como mecanismo secuencial de modelado temporal,
- un modelo basado en atención tipo Transformer como alternativa para capturar dependencias temporales de mayor alcance.

Se opta deliberadamente por no emplear pre-entrenamiento, *data augmentation* ni *learning rate scheduling*. Esta decisión implica que los rendimientos absolutos serán inferiores a los de los sistemas optimizados para competición (como el *baseline* VGG-ish de MediaEval, que alcanza ROC-AUC = 0,725 con 1000 épocas y umbrales optimizados), pero permite atribuir las diferencias observadas exclusivamente a la componente arquitectónica analizada: el mecanismo de agregación temporal. La transferencia de aprendizaje [31] y las técnicas de augmentación, aunque beneficiosas para el rendimiento absoluto, introducirían variables confundentes que dificultarían la interpretación del estudio comparativo.

El objetivo del proyecto no es optimizar el rendimiento absoluto ni reproducir configuraciones altamente especializadas orientadas a competición, sino analizar con rigor cómo cambia el rendimiento y el patrón de errores al introducir mecanismos temporales explícitos. El estudio se apoya en el subconjunto `autotagging_moodtheme` del MTG-Jamendo Dataset y en el *split-0* oficial por artista, garantizando comparabilidad y evitando *data leakage*. La evaluación se realiza con métricas estándar en MIR (ROC-AUC y PR-AUC/AP), complementadas con análisis por etiqueta y con procedimientos de interpretación estadística que permiten estudiar de forma más precisa el comportamiento real de cada arquitectura.

En suma, el problema que aborda este TFM puede sintetizarse en la siguiente pregunta: bajo un protocolo experimental controlado, en qué medida el modelado temporal explícito mejora (o no) el reconocimiento de etiquetas *mood/theme*, y si dicho efecto es consistente a nivel global o depende de la etiqueta considerada.

Capítulo 3

Objetivos

3.1. Objetivos generales

El objetivo general de este Trabajo Fin de Máster es analizar de forma comparativa el comportamiento de distintas arquitecturas de Deep Learning en el reconocimiento automático de emociones musicales a partir de audio, utilizando el subconjunto *mood/theme* del dataset MTG-Jamendo.

En particular, se pretende evaluar y contrastar tres familias de modelos — redes neuronales convolucionales (CNN), redes convolucionales recurrentes (CRNN) y arquitecturas CNN-Transformer — bajo un protocolo experimental controlado que mantiene constantes el conjunto de datos, las particiones oficiales, la representación de entrada y la función de pérdida. El propósito es evaluar de forma controlada el impacto del mecanismo de agregación temporal (ausente, recurrente o basado en atención) y analizar su efecto tanto a nivel global como por etiqueta emocional individual.

3.2. Objetivos específicos

En este trabajo se definen los siguientes objetivos específicos:

1. Analizar el estado del arte en el reconocimiento de emociones musicales, con especial atención a los trabajos basados en el dataset MTG-Jamendo y a las arquitecturas CNN, CRNN y modelos Transformer aplicados a *music auto-tagging*, identificando sus principales aportaciones y las limitaciones metodológicas que dificultan la comparabilidad entre estudios.

2. Establecer un marco experimental controlado y reproducible, basado en los *splits* oficiales del MTG-Jamendo Dataset por artista, representaciones de entrada consistentes (mel-espectrogramas precomputados de dimensión fija) y un protocolo de entrenamiento homogéneo, que permita atribuir las diferencias de rendimiento principalmente al tipo de modelado temporal, minimizando variaciones en los datos o en el pipeline.

3. Implementar tres modelos representativos de las principales familias de arquitecturas — CNN como *baseline* sin modelado temporal explícito, CRNN con modelado secuencial mediante LSTM, y CNN-Transformer con modelado basado en *self-attention* — compartiendo el mismo *backbone* convolucional para maximizar la comparabilidad.

4. Evaluar el rendimiento de los modelos mediante métricas estándar de clasificación multi-etiqueta empleadas en la literatura de MIR (ROC-AUC macro, PR-AUC/AP macro, F1 macro y micro), con el objetivo de obtener una visión global comparable con trabajos previos sobre el mismo benchmark.

5. Analizar el comportamiento de los modelos a nivel de etiqueta emocional individual (*mood/theme*), estudiando el rendimiento por etiqueta para identificar en qué etiquetas el modelado temporal muestra ventajas o desventajas y cuáles presentan dificultades sistemáticas independientes de la arquitectura.

6. Evaluar la estabilidad estadística de las diferencias observadas entre modelos, aplicando procedimientos de estimación de incertidumbre (intervalos de confianza por *bootstrap*) y tests de significación por etiqueta (test de DeLong para comparación de AUC), con el fin de distinguir diferencias consistentes de variaciones atribuibles al muestreo.

7. Analizar el impacto del umbral de decisión sobre las métricas dependientes de umbral (F1), evaluando la idoneidad del umbral fijo estándar ($t = 0{,}5$) en presencia de desbalance extremo entre etiquetas y la mejora obtenida mediante optimización de umbrales por etiqueta sobre el conjunto de validación.

8. Interpretar las limitaciones observadas en los modelos, relacionando los resultados con las características de las etiquetas (frecuencia, ambigüedad semántica, dependencia temporal), la naturaleza del benchmark y las restricciones del protocolo experimental adoptado.

3.3. Beneficios del proyecto

- **Evidencia empírica sobre el valor del modelado temporal en *auto-tagging* emocional.** El estudio aporta una comparación directa entre ausencia de modelado temporal, modelado secuencial y modelado por atención, bajo condiciones que permiten atribuir las diferencias observadas a la arquitectura. Esto complementa los hallazgos de Won et al. [2], que documentaron la dificultad de comparar arquitecturas cuando los factores experimentales no están controlados.

- **Marco experimental reproducible y reutilizable.** El uso de un dataset público con *splits* oficiales por artista, código desarrollado y un protocolo documentado permite que los resultados sirvan como referencia para estudios posteriores que quieran ampliar la comparación (por ejemplo, incorporando aumentación, pre-entrenamiento o arquitecturas adicionales) manteniendo la base experimental común.

- **Análisis por etiqueta que va más allá de las métricas agregadas.** En tareas multi-etiqueta desbalanceadas, las métricas globales pueden ocultar diferencias relevantes entre etiquetas. El análisis por etiqueta con validación estadística permite identificar qué emociones se benefician de forma consistente del contexto temporal y cuáles están limitadas por factores independientes de la arquitectura (escasez de ejemplos, ambigüedad semántica de la etiqueta, ausencia de correlato acústico claro).

- **Diagnóstico del efecto del umbral de decisión en métricas de clasificación.** El análisis de umbrales documenta cómo un umbral fijo estándar puede penalizar métricas como F1 en presencia de desbalance extremo, contribuyendo a una interpretación más informada de los resultados experimentales en este tipo de tareas.

Capítulo 4

Desarrollo del proyecto

4.1. Planificación del proyecto

El desarrollo del proyecto se ha estructurado en distintas fases, siguiendo una planificación orientada a garantizar la coherencia metodológica, la reproducibilidad experimental y la correcta alineación con los objetivos definidos. Dado el carácter académico y experimental del trabajo, se ha priorizado una planificación que combine el estudio teórico con el diseño riguroso del marco experimental y el análisis detallado de resultados.

Fase 1 (octubre–noviembre): Estudio del estado del arte y marco teórico.

En esta fase se analizó el panorama general del reconocimiento automático de emociones musicales a partir de audio junto a los principales enfoques de *Deep Learning* utilizados en la literatura. Se estudiaron las representaciones de entrada más habituales, en particular los mel-espectrogramas, así como los *datasets* empleados en trabajos previos. Este análisis permitió identificar el MTG-Jamendo Dataset como el conjunto de datos más adecuado para el desarrollo del proyecto y centrar el estudio en los enfoques más representativos utilizados sobre él.

Fase 2 (diciembre): Análisis técnico del dataset y diseño experimental.

Durante esta fase se trabajó directamente con el repositorio del MTG-Jamendo Dataset, replicando su estructura y explorando los datos disponibles con el objetivo de comprender en profundidad su organización, subconjuntos y *splits* oficiales. Este trabajo práctico permitió evaluar qué información estaba disponible, decidir el uso del subconjunto *mood/theme* y definir un marco experimental homogéneo que serviría como base para la implementación y comparación posterior de modelos.

Fase 3 (enero): Implementación de modelos y entrenamiento.

Durante esta fase se desarrolló la parte central del proyecto a nivel técnico. Se implementaron las tres arquitecturas de *Deep Learning* definidas en los objetivos — CNN, CRNN y CNN-Transformer— utilizando un mismo *pipeline* de procesamiento, entrenamiento y evaluación. Los modelos se entrenaron sobre los *splits* oficiales del *dataset* siguiendo la configuración definida en la fase anterior, generando los *checkpoints* y las predicciones necesarias para su posterior análisis.

Fase 4 (enero–febrero): Evaluación y análisis de resultados.

En esta fase se evaluó el rendimiento de los modelos entrenados mediante métricas estándar de clasificación multi-etiqueta. Además de la evaluación global, se realizó un análisis detallado del comportamiento por etiqueta emocional (*mood/theme*), se aplicaron procedimientos de validación estadística (intervalos de confianza por *bootstrap* y test de DeLong por etiqueta) y se llevó a cabo un estudio de optimización de umbrales de decisión. Asimismo, se generaron las figuras y tablas necesarias para la presentación de los resultados.

Fase 5 (febrero): Redacción final de la memoria.

Aunque durante el desarrollo del proyecto se han realizado entregas intermedias y avances parciales, la redacción final de la memoria se lleva a cabo una vez completado el trabajo experimental. En esta fase se realiza una revisión global del proyecto, integrando los resultados obtenidos y asegurando una presentación coherente y unificada del desarrollo, la metodología y las conclusiones.

De forma paralela al desarrollo de las distintas fases del proyecto, se han mantenido reuniones periódicas con el tutor, con una frecuencia aproximada de tres semanas. En estas sesiones se ha realizado el seguimiento del trabajo desarrollado, se han revisado los avances alcanzados y se han definido ajustes metodológicos, correcciones y líneas de mejora para las fases posteriores del proyecto.

4.2. Descripción de la solución, metodología y herramientas empleadas

4.2.1. Descripción de la solución

La solución desarrollada consiste en un sistema experimental para la evaluación comparativa de modelos de *Deep Learning* aplicados al reconocimiento automático de emociones musicales a partir de audio. Dicho sistema se estructura como un *pipeline* común que permite entrenar, evaluar y analizar distintas arquitecturas bajo un mismo conjunto de datos y un protocolo experimental único.

El punto de partida del sistema son mel-espectrogramas precomputados del MTG-Jamendo Dataset, que representan cada pista musical como una matriz tiempo-frecuencia. A partir de esta representación de entrada, el sistema produce, para cada canción del conjunto de test, un vector de *logits* que, tras aplicar la función *sigmoid*, se interpreta como un vector de probabilidades asociado a las 56 etiquetas *mood/theme*, permitiendo su evaluación mediante métricas de clasificación multi-etiqueta.

Para ello, se implementan e integran en un mismo *pipeline* tres familias de arquitecturas representativas del estado del arte:

- Una red neuronal convolucional (CNN), utilizada como modelo base de referencia, sin modelado temporal explícito.
- Una red convolucional recurrente (CRNN), que incorpora modelado temporal secuencial mediante una capa LSTM.
- Un modelo CNN-Transformer, que emplea mecanismos de *self-attention* para capturar dependencias temporales de forma paralela.

Los tres modelos comparten el mismo *backbone* convolucional, reciben la misma representación de entrada y se evalúan con las mismas métricas. El resultado final de la solución es un conjunto homogéneo de predicciones, métricas globales, métricas por etiqueta y análisis complementarios (validación estadística y optimización de umbrales) que permite analizar de forma sistemática cómo varía el comportamiento de cada arquitectura en función del tipo de emoción y del mecanismo de agregación temporal empleado.

4.2.2. Metodología

La metodología seguida en este proyecto se basa en el diseño de un *pipeline* experimental homogéneo que permite entrenar, evaluar y comparar las tres arquitecturas bajo las mismas condiciones. A continuación se detalla cada componente del *pipeline*.

Preparación de los datos y representación de entrada

Se utilizan mel-espectrogramas precomputados del MTG-Jamendo Dataset, almacenados en formato `.npy` con una resolución de $F = 96$ bandas mel y una dimensión temporal T variable según la duración de cada pista. Cada archivo se organiza en subdirectorios según el identificador de pista (`track_id%100`), siguiendo la estructura de directorios del *dataset*.

Para garantizar una entrada de dimensión constante, cada mel-espectrograma se ajusta a una ventana temporal fija de $T = 1366$ *frames* mediante recorte o relleno centrado (*center crop/pad*): si la pista supera los 1366 *frames*, se recorta simétricamente desde el centro; si es más corta, se rellena con ceros de forma simétrica. Este procesamiento es determinista e idéntico para todos los modelos.

La entrada resultante para cada pista tiene dimensión $(1, 96, 1366)$, donde el primer eje corresponde al canal (mono). Cada muestra se acompaña de un vector de etiquetas *multi-hot* de dimensión $(56,)$ con valores binarios (0,0 o 1,0), codificado según la lista de 56 etiquetas *mood/theme* en orden alfabético para garantizar un mapeo determinista y reproducible.

Las particiones de datos se cargan directamente desde los ficheros oficiales del *split-0* del subconjunto `autotagging_moodtheme` [1, 7]. Se verifica programáticamente la ausencia de solapamiento entre las particiones de entrenamiento, validación y test, así como la ausencia de duplicados internos, como medida explícita para reducir el riesgo de *data leakage*.

Protocolo de entrenamiento

Se implementa un protocolo de entrenamiento común para las tres arquitecturas, con los siguientes hiperparámetros fijos:

- **Función de pérdida:** `BCEWithLogitsLoss` (entropía cruzada binaria con *sigmoid* integrado, una pérdida por cada una de las 56 etiquetas).
- **Optimizador:** Adam con los parámetros por defecto de PyTorch (`weight_decay` = 0, sin *gradient clipping*).
- **Tasa de aprendizaje:** 1e-3 para CNN y CRNN; 1e-4 para CNN-Transformer. La reducción en el caso del Transformer se justifica empíricamente: con lr = 1e-3, el modelo diverge de forma catastrófica (la ROC-AUC de validación alcanza su máximo en la primera época y desciende por debajo del azar en la sexta). Esta observación es consistente con la sensibilidad de los Transformers a la tasa de aprendizaje documentada en la literatura [21, 23].
- **Tamaño de *batch*:** 16 para los tres modelos.
- **Máximo de épocas:** 100.
- ***Early stopping*:** se detiene el entrenamiento si la ROC-AUC macro sobre el conjunto de validación no mejora durante 5 épocas consecutivas (*patience* = 5). Se guarda el *checkpoint* del modelo con mejor ROC-AUC de validación.

- **Semilla:** 42 (fijada para `random`, `numpy`, `torch` y `torch.cuda/mps`, con `deterministic = True` y `benchmark = False` en cuDNN).

- **Sin técnicas adicionales:** no se emplea *data augmentation*, pre-entrenamiento, *learning rate scheduling* ni *weight decay*. Esta decisión es deliberada: permite atribuir las diferencias observadas principalmente al mecanismo de agregación temporal, reduciendo variables confundentes.

Los modelos se entrenan en el dispositivo disponible (GPU CUDA, Apple Silicon MPS o CPU), con gestión explícita de caché de memoria al finalizar cada época.

Arquitecturas implementadas

Backbone convolucional compartido. Las tres arquitecturas comparten un *backbone* convolucional idéntico cuya función es extraer características acústicas del mel-espectrograma. Este *backbone* está formado por 4 bloques, cada uno compuesto por:

$$\text{Conv2d}(3 \times 3) \;\rightarrow\; \text{BatchNorm2d} \;\rightarrow\; \text{ReLU} \;\rightarrow\; \text{MaxPool2d}(2 \times 2) \tag{4.1}$$

Cada bloque aplica una convolución 2D que detecta patrones locales en el dominio tiempo-frecuencia, seguida de una normalización por lotes (`BatchNorm2d`, que estabiliza el entrenamiento), una activación no lineal (ReLU) y un *max-pooling* que reduce la resolución a la mitad en ambas dimensiones. Los filtros progresan como $1 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 512$, incrementando la capacidad representacional en cada bloque.

Tras los cuatro bloques de *pooling*, la salida tiene dimensión $(\text{batch}, 512, 6, 85)$: 512 mapas de características, cada uno de tamaño 6 (lo que queda de las 96 bandas mel tras cuatro reducciones a la mitad) $\times$ 85 (lo que queda de los 1366 *frames* temporales). En este punto, la dimensión de frecuencia está muy comprimida (6 *bins*), mientras que la dimensión temporal (85 pasos) aún preserva la estructura temporal de la pieza. Esta asimetría es lo que motiva las distintas estrategias de agregación temporal en cada modelo.

Compartir el *backbone* entre las tres arquitecturas es una decisión clave del diseño experimental: significa que las características acústicas de partida son idénticas, y la diferencia principal entre modelos es cómo se agrega la información temporal a partir de ese punto. En la implementación, tanto la CRNN como el CNN-Transformer instancian internamente un objeto `CNNBaseline` y extraen sus capas convolucionales. Dado que se fija la semilla antes de la creación de cada modelo, los pesos del *backbone* se inicializan de forma idéntica en los tres casos (si bien cada modelo entrena su propia copia de forma independiente durante el entrenamiento).

Los tres modelos producen como salida un vector de *logits* sin activación (*raw logits*), es decir, valores reales que no han pasado por una función *sigmoid*. Esto se debe a que la función de pérdida utilizada (`BCEWithLogitsLoss`) ya incorpora el *sigmoid* internamente por estabilidad numérica. En fase de evaluación, se aplica *sigmoid* por separado para obtener probabilidades en el rango $[0, 1]$.

A continuación se describe cada arquitectura, indicando paso a paso cómo transforma la salida del *backbone* hasta producir las 56 predicciones.

CNN Baseline. La CNN es el modelo más sencillo de los tres y sirve como línea base de referencia. No incorpora ningún mecanismo explícito de modelado temporal: la información temporal y frecuencial se agrega de forma simétrica, sin distinguir entre ambas dimensiones.

- **Entrada:** salida del *backbone*, dimensión $(\text{batch}, 512, 6, 85)$.
- ***Global Average Pooling* (GAP):** se calcula el promedio de cada uno de los 512 mapas de características sobre todas las posiciones (6 frecuencias $\times$ 85 tiempos), colapsando ambas dimensiones en un único valor por mapa. El resultado es un vector de 512 componentes por muestra. Esta operación trata la frecuencia y el tiempo de forma simétrica: no distingue si un patrón ocurre al principio, al final o en una banda de frecuencia concreta.
- ***Dropout*** ($p = 0{,}5$): durante el entrenamiento, se desactivan aleatoriamente el 50 % de las neuronas para reducir el sobreajuste.
- **Capa lineal** ($512 \rightarrow 56$): proyección final que produce un *logit* por cada etiqueta *mood/theme*.
- **Salida:** vector de 56 *logits* (sin *sigmoid*).

Parámetros totales: 1.580.472.

CRNN. La CRNN parte de la misma base convolucional que la CNN, pero introduce un cambio fundamental: en lugar de colapsar la dimensión temporal junto con la frecuencia, la preserva y la procesa secuencialmente con una red recurrente (LSTM). La hipótesis que motiva este diseño es que la evolución temporal de las características acústicas (por ejemplo, transiciones de intensidad, cambios de textura o estructura musical a lo largo de la canción) puede ser relevante para identificar ciertas emociones.

- **Entrada:** salida del *backbone*, dimensión $(\text{batch}, 512, 6, 85)$.

- **Colapso de frecuencia:** se promedia la dimensión de frecuencia (tamaño 6), obteniendo (batch, 512, 85). Esto destruye la información frecuencial detallada pero preserva la secuencia temporal de 85 pasos. Se toma esta decisión porque, tras cuatro capas de *pooling*, los 6 *bins* restantes ya representan información frecuencial muy comprimida, y lo relevante para el modelado temporal es cómo evolucionan las características a lo largo del tiempo.

- **Reorganización:** se transponen las dimensiones para pasar de (batch, 512, 85) a (batch, 85, 512), es decir, una secuencia de 85 pasos temporales donde cada paso tiene 512 características. Este formato es el que espera la LSTM.

- **LSTM (*Long Short-Term Memory*):** una capa LSTM unidireccional (`input_size` = 512, `hidden_size` = 256) procesa la secuencia paso a paso, de izquierda a derecha. En cada paso temporal, la LSTM combina la entrada actual con una memoria interna que acumula información de los pasos anteriores, permitiendo capturar dependencias temporales. La salida es una secuencia de 85 vectores de 256 dimensiones, donde cada vector incorpora información del contexto temporal previo.

- ***Mean pooling* temporal:** se promedian los 85 vectores de salida de la LSTM en un único vector de 256 componentes. Esto agrega la información temporal procesada por la LSTM en una representación fija.

- ***Dropout*** ($p = 0{,}5$) y **capa lineal** ($256 \rightarrow 56$): regularización y proyección final a 56 *logits*.

- **Salida:** vector de 56 *logits* (sin *sigmoid*).

Parámetros totales: 2.354.616.

CNN-Transformer. El CNN-Transformer comparte con la CRNN el tratamiento inicial (colapso de frecuencia y preservación temporal), pero sustituye la LSTM por un *Transformer Encoder*. Si la LSTM procesa la secuencia paso a paso de forma secuencial, el Transformer utiliza un mecanismo de *self-attention* [21] que permite a cada paso temporal «atender» directamente a todos los demás de forma simultánea. En principio, esto facilita la captura de dependencias de largo alcance sin verse limitado por la naturaleza secuencial de la recurrencia.

- **Entrada:** salida del *backbone*, dimensión (batch, 512, 6, 85).

- **Colapso de frecuencia y reorganización:** idéntico a la CRNN. Se obtiene una secuencia de 85 pasos con 512 características cada uno.

- **Proyección lineal** ($512 \rightarrow 256$): reduce la dimensionalidad de cada paso temporal de 512 a $d_{\text{model}} = 256$ para adaptarse a la configuración del Transformer y limitar el coste computacional.

- **Codificación posicional aprendible:** a diferencia de la LSTM, que conoce implícitamente el orden de la secuencia (la procesa de izquierda a derecha), el Transformer en sí mismo no tiene noción del orden temporal: trata la entrada como un conjunto, no como una secuencia. Para subsanar esto, se suma a cada posición un vector aprendible que codifica su posición en la secuencia (85 posiciones, inicializadas como ruido gaussiano $\times$ 0,02). De este modo, el modelo puede distinguir si un patrón ocurre al principio, en medio o al final de la pieza.

- ***Transformer Encoder*:** 2 capas de codificación, cada una con un bloque de *self-attention* de 4 cabezas (*head_dim* = 64) y una red *feedforward* interna de dimensión 512, con *dropout* = 0,5. La *self-attention* calcula, para cada paso temporal, un promedio ponderado de todos los pasos de la secuencia, donde los pesos de atención se aprenden durante el entrenamiento. Esto permite al modelo relacionar directamente momentos distantes del audio sin pasar por todos los pasos intermedios, como tendría que hacer una LSTM.

- ***Mean pooling* temporal:** se promedian los 85 vectores de salida del *encoder* en un único vector de 256 componentes.

- ***Dropout*** (p = 0,5) y **capa lineal** (256 $\rightarrow$ 56): regularización y proyección final a 56 *logits*.

- **Salida:** vector de 56 *logits* (sin *sigmoid*).

Parámetros totales: 2.773.432.

Tabla 4.1: Resumen comparativo de las tres arquitecturas implementadas.

Aspecto	CNN	CRNN	CNN-Transformer
Backbone CNN	Compartido	Compartido	Compartido
Agregación temporal	Simétrica (GAP)	Secuencial (LSTM)	Paralela (*self-attention*)
Dimensión cabezal	512	256	256
Parámetros	1.580.472	2.354.616	2.773.432

Resumen comparativo de las arquitecturas. Las tres arquitecturas se diferencian principalmente en cómo tratan la dimensión temporal tras el *backbone*: la CNN la destruye por promedio junto con la frecuencia; la CRNN la preserva y la procesa secuencialmente; y el CNN-Transformer la preserva y la procesa en paralelo mediante atención. Esta es la variable que el estudio pretende evaluar bajo condiciones controladas.

Evaluación sobre el conjunto de test

Una vez entrenados los modelos, se evalúan exclusivamente sobre el conjunto de test del *split-0* (4.231 muestras). Los *checkpoints* guardados durante el entrenamiento se cargan en modo evaluación (`model.eval()`), sin reentrenar ni modificar los pesos.

Para cada pista del test se obtiene el vector de *logits*, se aplica la función *sigmoid* para obtener probabilidades en $[0, 1]$ y se genera una predicción binaria aplicando un umbral fijo $t = 0{,}5$.

Se calculan las siguientes métricas mediante `sklearn`:

- **ROC-AUC macro:** media aritmética de la ROC-AUC de cada etiqueta. Métrica principal de *ranking*, independiente del umbral.
- **PR-AUC / *Average Precision* macro (mAP):** media de la precisión promedio por etiqueta. Más exigente que ROC-AUC en presencia de desbalance.
- **F1 macro:** media del F1-*score* por etiqueta, con umbral fijo $t = 0{,}5$. Métrica dependiente del umbral.
- **F1 micro:** F1-*score* calculado sobre todas las predicciones agregadas.

Adicionalmente, se calculan ROC-AUC y *Average Precision* de forma independiente para cada una de las 56 etiquetas, lo que permite el análisis por etiqueta emocional individual.

Análisis complementarios

Además de la evaluación directa sobre test, se realizan dos análisis complementarios:

1. **Optimización de umbrales de decisión.** Para cada uno de los 56 *labels* y cada modelo, se evalúa el F1-*score* sobre el conjunto de validación para 99 umbrales (de 0,01 a 0,99, con paso 0,01). Se selecciona el umbral que maximiza el F1 por etiqueta en validación y se aplica al conjunto de test. Se recalculan F1 macro y micro con los umbrales optimizados.

   Este análisis permite cuantificar el impacto del umbral fijo estándar ($t = 0{,}5$) en presencia de desbalance extremo entre etiquetas.

2. **Validación estadística.** Se implementan dos procedimientos:

   - *Bootstrap* (1.000 remuestreos con reemplazo del test): para cada remuestreo se calcula la ROC-AUC macro de cada modelo. Se obtienen intervalos de confianza al 95 % (percentiles 2,5 y 97,5) y se calculan deltas por pares con una estimación empírica basada en *bootstrap* (proporción de remuestreos en los que la diferencia cambia de signo).
   - Test de DeLong [32]: comparación de AUC por etiqueta entre cada par de modelos. Se excluyen etiquetas con menos de 5 positivos o 5 negativos en test. Se obtiene un z-estadístico y un p-valor bilateral por etiqueta.

Estos procedimientos permiten evaluar si las diferencias observadas entre modelos son estadísticamente consistentes o atribuibles al azar del muestreo.

4.2.3. Herramientas empleadas

El desarrollo del proyecto se ha realizado en Python, utilizando las siguientes herramientas principales:

- **PyTorch:** *framework* de *Deep Learning* utilizado para la implementación de las arquitecturas, la definición de los ciclos de entrenamiento, el cálculo de la función de pérdida (`BCEWithLogitsLoss`) y la gestión de dispositivos de cómputo (CUDA/MPS/CPU).
- **NumPy:** manipulación de mel-espectrogramas (`.npy`), operaciones de *crop/pad* y codificación *multi-hot* de etiquetas.
- **pandas:** lectura de ficheros TSV de *splits* y etiquetas del *dataset*, y generación de tablas de resultados.
- **scikit-learn:** cálculo de métricas de evaluación (`roc_auc_score`, `average_precision_score`, `f1_score`).
- **SciPy:** funciones estadísticas utilizadas en el test de DeLong y en el cálculo de p-valores.
- **Matplotlib y Seaborn:** generación de figuras de publicación (curvas de aprendizaje, *dot charts*, *heatmaps*, *scatter plots*) a 300 DPI.
- **python-dotenv:** gestión de rutas de datos mediante variables de entorno (`.env`), separando la configuración de *paths* del código fuente.

El desarrollo y la organización del código se han llevado a cabo mediante el entorno de desarrollo PyCharm, que ha permitido estructurar el proyecto en módulos (`src/data`, `src/models`), facilitar la depuración y gestionar las distintas fases de implementación.

Los experimentos se han ejecutado en un entorno macOS con aceleración por GPU mediante Apple Silicon (MPS *backend* de PyTorch). El código es compatible también con GPU NVIDIA (CUDA) y con ejecución en CPU, detectando automáticamente el dispositivo disponible.

4.3. Recursos requeridos

Para la ejecución del presente proyecto se han empleado los siguientes recursos:

Recursos técnicos y software

- Ordenador personal con procesador Apple Silicon y GPU integrada, utilizado tanto para el desarrollo del código como para el entrenamiento de los modelos.
- Sistema operativo macOS.
- Entorno de desarrollo PyCharm.
- Python 3.10 con entorno virtual dedicado.
- *Framework* de *Deep Learning* PyTorch con soporte MPS.
- Librerías auxiliares: NumPy, pandas, scikit-learn, SciPy, Matplotlib, Seaborn, python-dotenv.
- Mel-espectrogramas precomputados del MTG-Jamendo Dataset en formato `.npy` (varios GB de almacenamiento).
- Metadatos y *splits* oficiales del MTG-Jamendo Dataset.

Recursos bibliográficos y científicos

- Artículos científicos y publicaciones académicas relacionadas con el reconocimiento de emociones musicales y *music auto-tagging*, entre ellos:
  - Bogdanov et al. [7]: MTG-Jamendo Dataset for Automatic Music Tagging.
  - Bogdanov et al. [1]: MediaEval 2019 Emotion and Theme Recognition in Music Task.
  - Choi et al. [11]: Automatic Tagging Using Deep Convolutional Neural Networks.
  - Choi et al. [20]: Convolutional Recurrent Neural Networks for Music Classification.
  - Won et al. [2]: Evaluation of CNN-based Automatic Music Tagging Models.

- Documentación técnica de PyTorch, scikit-learn y el repositorio oficial del MTG-Jamendo Dataset.

Recursos humanos

- Tutor académico del Trabajo Fin de Máster, para la supervisión y orientación del proyecto.

4.4. Presupuesto

En este apartado se presenta la estimación económica del proyecto, desglosada en costes de personal, equipamiento, software y otros costes directos.

Tabla 4.2: Estimación del presupuesto del proyecto.

Tipo de coste	**Valor (€)**	**Comentarios**
Horas de trabajo	3,000,00	150 horas valoradas a 20 €/h (tarifa de referencia para perfil técnico júnior)
Equipo técnico utilizado	1,000,00	Ordenador personal utilizado para desarrollo y ejecución de experimentos: MacBook Air (Apple Silicon M4, 16 GB RAM)
Software utilizado	0,00	Software y librerías con licencia gratuita / código abierto (Python, PyTorch, scikit-learn, etc.)
Estudios e informes	0,00	No se adquirieron informes de pago. Artículos y documentación consultados mediante recursos abiertos o institucionales
Materiales empleados	0,00	No se utilizaron materiales de laboratorio, sensores ni hardware adicional
Total estimado	**4,000,00**	

4.5. Viabilidad

A continuación se presenta un breve análisis de la viabilidad económica del proyecto y de la sostenibilidad de sus resultados.

Viabilidad económica

El coste total estimado del proyecto (4,000 €) es reducido en comparación con trabajos de investigación equivalentes en el ámbito del *Deep Learning* aplicado a audio. Se explica por tres factores principales:

- Todo el *stack* tecnológico utilizado (Python, PyTorch, scikit-learn y el resto de librerías) es de código abierto y distribución gratuita, lo que elimina por completo los costes de licencias de software.

- El *dataset* utilizado (MTG-Jamendo) es un recurso público y gratuito, con mel-espectrogramas precomputados disponibles para descarga directa. No ha sido necesario adquirir datos de pago ni contratar servicios de anotación.

- Los experimentos se han ejecutado íntegramente en un equipo personal con GPU integrada (Apple Silicon), sin requerir servicios de computación en la nube ni infraestructura de GPU dedicada. La escala de los modelos implementados (entre 1,5 y 2,7 millones de parámetros) y la duración del entrenamiento (entre 15 y 30 épocas, con tiempos del orden de minutos por época) hacen viable su ejecución en *hardware* de consumo.

En relación coste/beneficio, el proyecto produce un marco experimental completo — código, modelos entrenados, predicciones, métricas por etiqueta y análisis estadísticos— con una inversión modesta, donde la práctica totalidad del coste corresponde a horas de trabajo intelectual.

Sostenibilidad del proyecto

Desde el punto de vista de la sostenibilidad, el proyecto se apoya íntegramente en código abierto, datos públicos y *hardware* de consumo, lo que permite replicar y extender los experimentos sin coste adicional en licencias, datos ni infraestructura. El *pipeline* está diseñado de forma modular, de modo que incorporar nuevas arquitecturas o técnicas requiere intervenir únicamente en el módulo del modelo. Las dependencias tecnológicas (PyTorch, scikit-learn, NumPy, SciPy) son proyectos maduros con mantenimiento activo, y los resultados obtenidos constituyen una línea base documentada reutilizable en trabajos posteriores. Adicionalmente, la escala moderada de los modelos permite completar el entrenamiento en pocas horas sobre *hardware* convencional, con un consumo energético reducido frente a enfoques basados en pre-entrenamiento masivo.

En conjunto, el proyecto es económicamente viable con recursos propios de un estudiante de máster, y sus resultados son reutilizables y ampliables sin inversión adicional significativa.

4.6. Resultados del proyecto

En esta sección se presentan los resultados obtenidos en el desarrollo del proyecto, organizados en dos bloques: los resultados del desarrollo (qué se implementó y qué artefactos se produjeron) y los resultados experimentales (métricas de rendimiento, análisis por etiqueta y validación estadística).

4.6.1. Resultados del desarrollo

El proyecto ha producido un sistema experimental completo y funcional para la evaluación comparativa de modelos de *Deep Learning* aplicados al reconocimiento de emociones musicales. Los principales componentes desarrollados son los siguientes:

- ***Pipeline* de datos.** Conjunto de módulos para la carga de mel-espectrogramas del MTG-Jamendo Dataset, la codificación *multi-hot* de las 56 etiquetas *mood/theme*, la aplicación de la ventana temporal fija (1366 *frames* mediante *center crop/pad*) y la construcción de los cargadores de datos (`DataLoaders`) de PyTorch. El *pipeline* incluye una verificación programática que comprueba la ausencia de solapamiento entre las particiones de entrenamiento, validación y test antes de comenzar cualquier experimento.

- **Tres arquitecturas de *Deep Learning*.** Implementación de los tres modelos definidos en los objetivos — CNN Baseline, CRNN y CNN-Transformer— compartiendo el mismo *backbone* convolucional de 4 bloques, tal como se detalla en la apartado 4.2.2.

- **Protocolo de entrenamiento automatizado.** Un *script* de entrenamiento por arquitectura con el ciclo completo: optimizador Adam, función de pérdida `BCEWithLogitsLoss`, *early stopping* con *patience* 5 sobre la ROC-AUC de validación, y guardado automático de los pesos del modelo con mejor rendimiento en validación (*checkpoint*).

- **Evaluación sobre test.** *Script* de evaluación que carga los *checkpoints* guardados durante el entrenamiento, genera las predicciones sobre el conjunto de test y calcula las métricas globales y por etiqueta.

- **Análisis complementarios.** Módulos de optimización de umbrales de decisión por etiqueta sobre el conjunto de validación y de validación estadística (*bootstrap* con intervalos de confianza y test de DeLong por etiqueta).

- **Figuras y tablas de publicación.** Generación automatizada de todas las figuras y tablas incluidas en esta memoria, a resolución de 300 DPI.

Adicionalmente, se conservan los registros del entrenamiento fallido del CNN-Transformer con lr = 1e-3 (descrito en la apartado 4.6.6), que sirven como evidencia documentada del comportamiento de divergencia del Transformer bajo la tasa de aprendizaje estándar.

4.6.2. Resultados experimentales

A continuación se presentan las métricas de rendimiento obtenidas por los tres modelos sobre el conjunto de test del *split-0* (4.231 muestras). Todos los resultados corresponden a los *checkpoints* con mejor ROC-AUC de validación, cargados en modo evaluación sin modificar los pesos.

Comparación global

La cuadro 4.3 resume las métricas principales obtenidas en test.

Tabla 4.3: Comparación del rendimiento en test de las tres arquitecturas. F1 calculado con umbral fijo $t = 0{,}5$. ROC-AUC y mAP son independientes del umbral. La columna «Mejor ép.» indica la época del mejor *checkpoint* sobre el total de épocas entrenadas.

Modelo	Params	ROC-AUC	mAP	F1 macro	F1 micro	Mejor ép.
CNN Baseline	1,58M	0,7022	0,0652	0,0107	0,0304	19/24
CRNN	2,35M	0,6988	0,0639	0,0111	0,0327	25/30
CNN-Transformer	2,77M	0,7029	0,0595	0,0048	0,0094	10/15

Los tres modelos alcanzan una ROC-AUC macro prácticamente idéntica, con una diferencia máxima de 0,41 puntos porcentuales entre el mejor (CNN-Transformer, 0,7029) y el peor (CRNN, 0,6988). En mAP macro, la CNN obtiene el valor más alto (0,0652), seguida de la CRNN (0,0639) y el CNN-Transformer (0,0595). Los valores absolutos de mAP, modestos en todos los casos, reflejan la dificultad intrínseca de la clasificación multi-etiqueta con 56 etiquetas bajo un desbalance severo de clases.

Las métricas de F1 con umbral fijo $t = 0{,}5$ son extremadamente bajas (F1 macro entre 0,005 y 0,011). Este resultado no indica que los modelos no hayan aprendido — la ROC-AUC superior a 0,70 demuestra que sí discriminan— sino que el umbral de 0,5 es inadecuado para la distribución de etiquetas del *dataset*, donde 26 de las 56 etiquetas tienen una frecuencia inferior al 1 %. Este punto se analiza en detalle en el apartado de optimización de umbrales.

Eficiencia de entrenamiento y generalización

La cuadro 4.4 compara la eficiencia y el comportamiento de generalización de los tres modelos.

Tabla 4.4: Eficiencia de entrenamiento y generalización. La columna «Val→Test» indica la diferencia entre la ROC-AUC en test y la ROC-AUC en validación, en puntos porcentuales: valores positivos indican que el modelo mejora al pasar a datos no vistos, valores negativos indican lo contrario.

Modelo	Params	Tiempo/ép.	Total	Mejor ép.	Val AUC	Test AUC	Val→Test
CNN Baseline	1,58M	9,7 min	234 min	19/24	0,6987	0,7022	+0,35 pp
CRNN	2,35M	9,4 min	283 min	25/30	0,7092	0,6988	−1,04 pp
CNN-Transformer	2,77M	7,1 min	107 min	10/15	0,6913	0,7029	+1,16 pp

Tres observaciones destacan de esta tabla:

- La CRNN es el único modelo que presenta una transferencia negativa de validación a test (−1,04 pp): obtiene la mejor ROC-AUC en validación (0,7092) pero la peor en test (0,6988). Este patrón es consistente con la hipótesis de un posible sobreajuste de la LSTM a patrones temporales específicos de los artistas del conjunto de validación que no generalizan a artistas no vistos en test. Cabe recordar que el MTG-Jamendo utiliza particiones por artista: los artistas de validación y test son completamente disjuntos, por lo que un modelo que memoriza patrones temporales propios de ciertos artistas puede rendir bien en validación pero peor al enfrentarse a artistas diferentes.

- El CNN-Transformer converge sustancialmente antes (mejor época 10 de 15, frente a 19/24 de CNN y 25/30 de CRNN), con el menor tiempo total de entrenamiento (107 minutos frente a 234 y 283). A pesar de tener la ROC-AUC de validación más baja (0,6913), obtiene la más alta en test (0,7029), mostrando la transferencia positiva más pronunciada (+1,16 pp). La convergencia temprana puede actuar como una forma implícita de regularización, limitando la exposición del modelo a los datos de entrenamiento.

- La CNN presenta un comportamiento equilibrado: convergencia intermedia (época 19), transferencia moderada (+0,35 pp) y la mayor mAP de los tres modelos.

La figura 4.1 muestra las curvas de aprendizaje de los tres modelos.

La conclusión principal de esta figura es el contraste entre ambos paneles: mientras la pérdida de entrenamiento desciende de forma continua en los tres modelos, la pérdida de validación oscila sin tendencia clara en una banda estrecha (0,082–0,088). Este patrón es un indicador clásico de sobreajuste incipiente y justifica la intervención del *early stopping*.

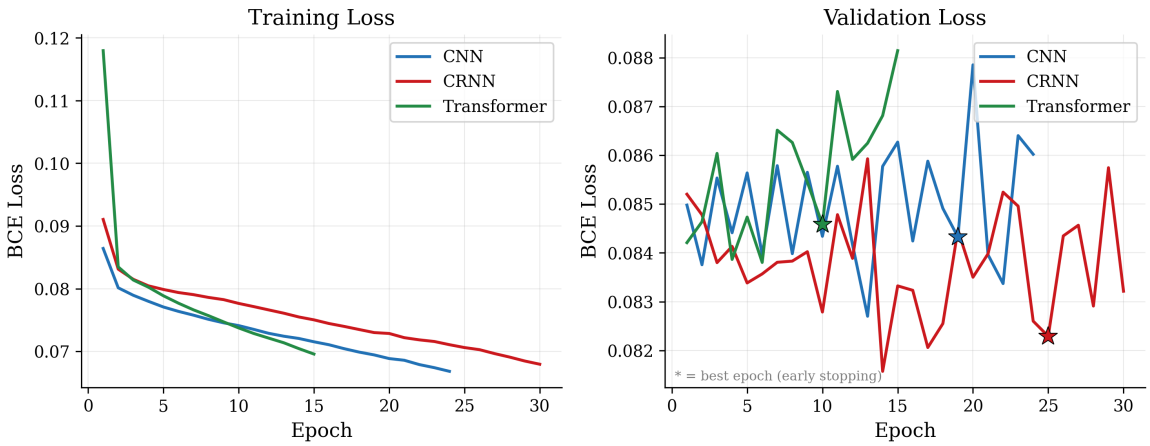

Figura 4.1: Curvas de aprendizaje de los tres modelos. Izquierda: pérdida de entrenamiento (BCE Loss) por época. Derecha: pérdida de validación por época. La estrella indica la mejor época (*early stopping*).

La figura 4.2 muestra la evolución de la ROC-AUC de validación a lo largo del entrenamiento.

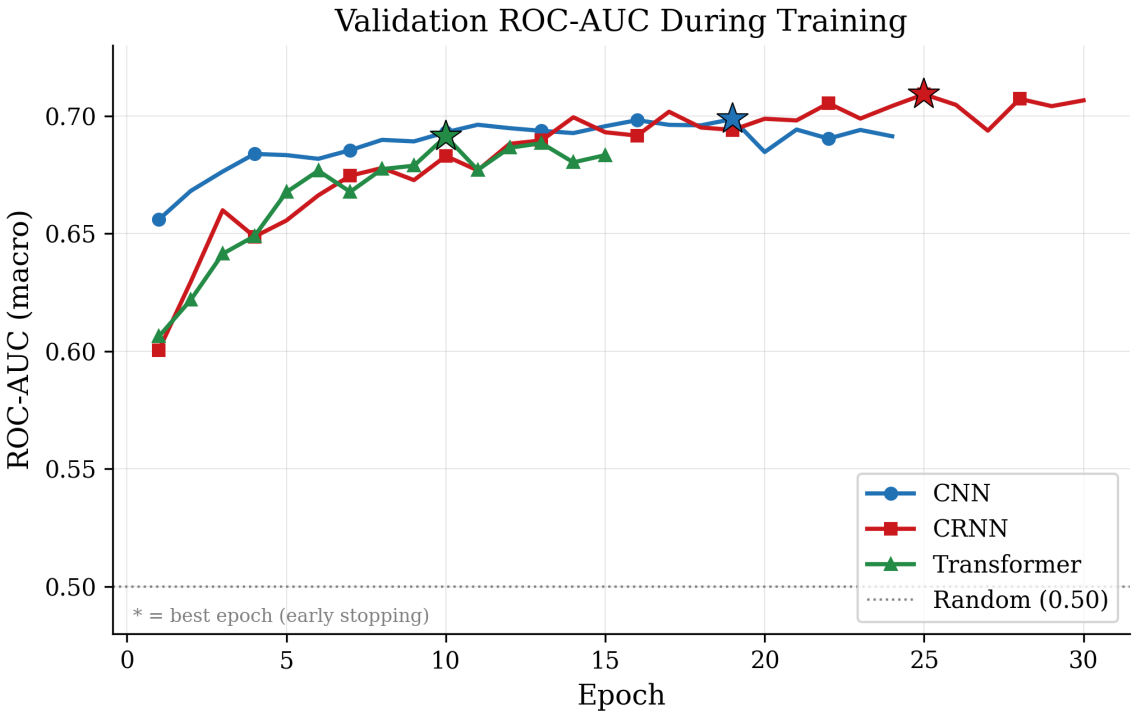

Figura 4.2: Evolución de la ROC-AUC macro de validación durante el entrenamiento. La línea punteada indica el nivel del azar (0,50). La estrella marca la mejor época de cada modelo.

Lo más relevante de esta figura es que, a pesar de que cada modelo alcanza su mejor época en momentos muy distintos (CNN en la 19, CRNN en la 25, Transformer en la 10), a partir de la época 8–10 las tres curvas convergen en una banda estrecha entre 0,68 y 0,71. Esto confirma visualmente que las diferencias finales entre arquitecturas son pequeñas y que las épocas adicionales no aportan mejoras en generalización.

La figura 4.3 muestra la relación entre el número de parámetros de cada modelo y su rendimiento en test (el tamaño del punto representa el tiempo total de entrenamiento).

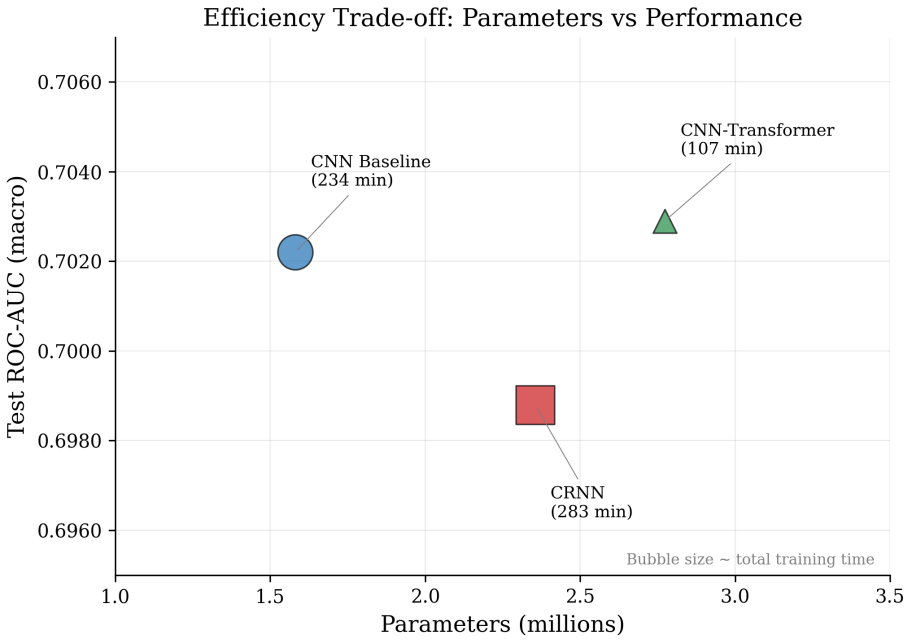

Figura 4.3: Compromiso entre número de parámetros y rendimiento en test (ROC-AUC macro). El tamaño de cada punto es proporcional al tiempo total de entrenamiento.

Este gráfico evidencia que no existe relación directa entre complejidad y rendimiento: la CNN (1,58M parámetros) alcanza una ROC-AUC prácticamente idéntica a la del CNN-Transformer (2,77M), y la CRNN (2,35M) obtiene la peor ROC-AUC con el mayor tiempo de entrenamiento (283 min). Este resultado sugiere que, bajo este protocolo, no existe una relación directa entre complejidad y rendimiento, y que parte de la limitación puede venir de las señales disponibles en el *dataset*.

Optimización de umbrales de decisión

En un sistema de clasificación multi-etiqueta, el modelo no responde directamente «sí» o «no» para cada etiqueta, sino que produce una probabilidad (por ejemplo, 0,12 para «happy» o 0,03 para «dark»). Para convertir esas probabilidades en decisiones binarias es necesario fijar un umbral: si la probabilidad supera el umbral, se asigna la etiqueta; si no, no se asigna. El umbral estándar es 0,5, pero en un *dataset* con desbalance extremo — donde 26 de las 56 etiquetas aparecen en menos del 1 % de las pistas— las probabilidades predichas son naturalmente bajas, y un umbral de 0,5 resulta demasiado exigente: el modelo prácticamente nunca lo supera, lo que explica que 54 de las 56 etiquetas obtengan F1 = 0 con ese umbral.

Para determinar si esta situación es un artefacto del umbral o una limitación real de los modelos, se realizó una optimización de umbrales por etiqueta. El procedimiento consiste en probar, para cada etiqueta y cada modelo, 99 umbrales distintos (de 0,01 a 0,99) sobre el conjunto de validación, seleccionando el que maximiza el F1.

A continuación, se aplican esos umbrales al conjunto de test para recalcular las métricas. De este modo, los umbrales no se ajustan sobre los datos de test, evitando cualquier sesgo.

Tabla 4.5: Impacto de la optimización de umbrales sobre las métricas de F1. La columna «Labels F1=0» indica el número de etiquetas (de 56) para las que el modelo no predice ningún positivo con el umbral indicado. Mediana del umbral óptimo: CNN 0,04; CRNN 0,04; Transformer 0,07.

Modelo	**F1 macro** (t=0,5)	**F1 macro** (optim.)	**F1 micro** (t=0,5)	**F1 micro** (optim.)	**Labels F1=0** (t=0,5)	**Labels F1=0** (optim.)
CNN Baseline	0,011	0,081 (×8)	0,030	0,094	54	9
CRNN	0,011	0,079 (×7)	0,033	0,105	54	12
CNN-Transformer	0,005	0,076 (×16)	0,009	0,102	54	14

La optimización produce mejoras de entre 7 y 16 veces en F1 macro, y reduce el número de etiquetas sin predicciones positivas de 54 a entre 9 y 14. Los umbrales óptimos son muy inferiores a 0,5 (la mediana se sitúa entre 0,04 y 0,07), lo cual refleja directamente el desbalance extremo de las etiquetas: cuando una emoción aparece en menos del 1 % de las pistas, la probabilidad predicha por el modelo es naturalmente baja, y solo un umbral igualmente bajo permite capturarla.

Este resultado confirma que los valores de F1 reportados con umbral fijo no reflejan la capacidad discriminativa real de los modelos — que se aprecia en la ROC-AUC, que es independiente del umbral— sino la inadecuación del umbral estándar para *datasets* con desbalance extremo.

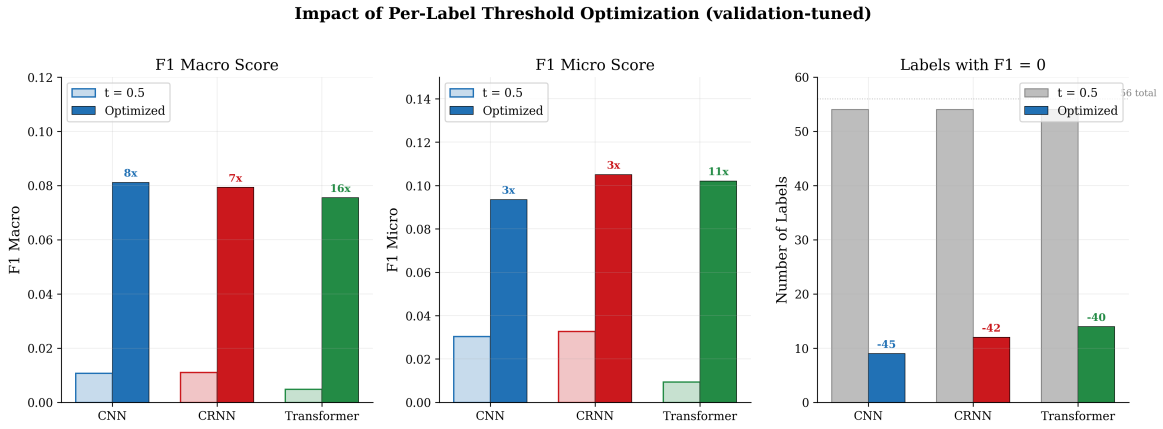

Figura 4.4: Impacto de la optimización de umbrales por etiqueta. Izquierda: F1 macro con umbral fijo (t = 0,5) frente a umbrales optimizados. Centro: F1 micro. Derecha: número de etiquetas con F1 = 0. Los multiplicadores indican la mejora relativa.

El contraste entre las barras de la izquierda (umbral fijo) y las de la derecha (umbrales optimizados) es la idea central: con un umbral adecuado por etiqueta, el F1 macro se multiplica entre 7 y 16 veces, y el número de etiquetas que el modelo es incapaz de predecir baja de 54 a menos de 15. Es decir, los modelos sí han aprendido; el umbral de decisión fijo (t = 0,5) era inadecuado para este desbalance, lo que afecta de forma directa a F1.

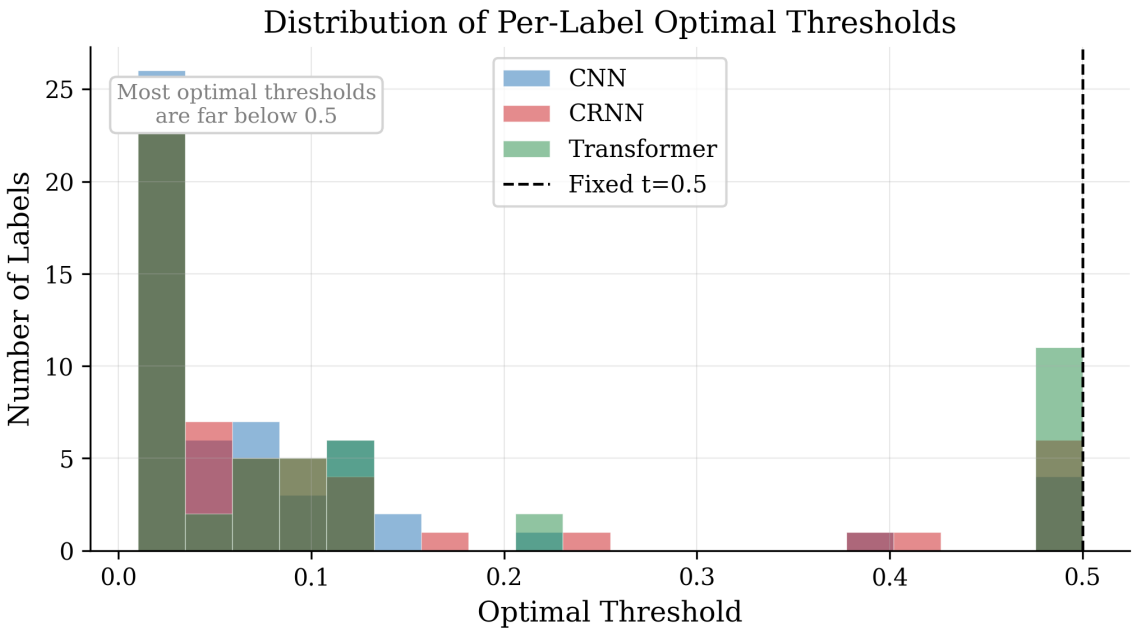

Figura 4.5: Distribución de los umbrales óptimos por etiqueta para cada modelo. La línea discontinua marca el umbral fijo estándar ($t = 0{,}5$). La mayoría de los umbrales óptimos se sitúan por debajo de 0,10.

La mayoría de los umbrales óptimos se concentran por debajo de 0,10, muy lejos del 0,5 estándar (línea discontinua). Esto refleja directamente el desbalance del *dataset*: como la mayoría de las emociones aparecen en muy pocas pistas, las probabilidades que el modelo asigna son bajas, y solo un umbral igualmente bajo permite capturarlas.

4.6.3. Análisis por etiqueta emocional

Las métricas globales (ROC-AUC macro, mAP macro) resumen el rendimiento de cada modelo con un solo número, promediando los resultados de las 56 etiquetas. Esto es útil para una comparación rápida, pero puede ocultar diferencias importantes: si un modelo mejora en 20 etiquetas y empeora en otras 20, el promedio no cambia. Por ello, se analizó el rendimiento de cada modelo etiqueta por etiqueta, lo que permite identificar en qué tipos de emociones cada arquitectura funciona mejor y formular hipótesis sobre las posibles causas.

Para contextualizar este análisis, la figura 4.6 muestra la distribución de frecuencias de las 56 etiquetas en el conjunto de entrenamiento. El desbalance es muy pronunciado: 26 de las 56 etiquetas aparecen en menos del 1 % de las pistas, mientras que unas pocas como «dark» o «electronic» superan el 5 %. Este desbalance condiciona directamente el rendimiento de los modelos por etiqueta.

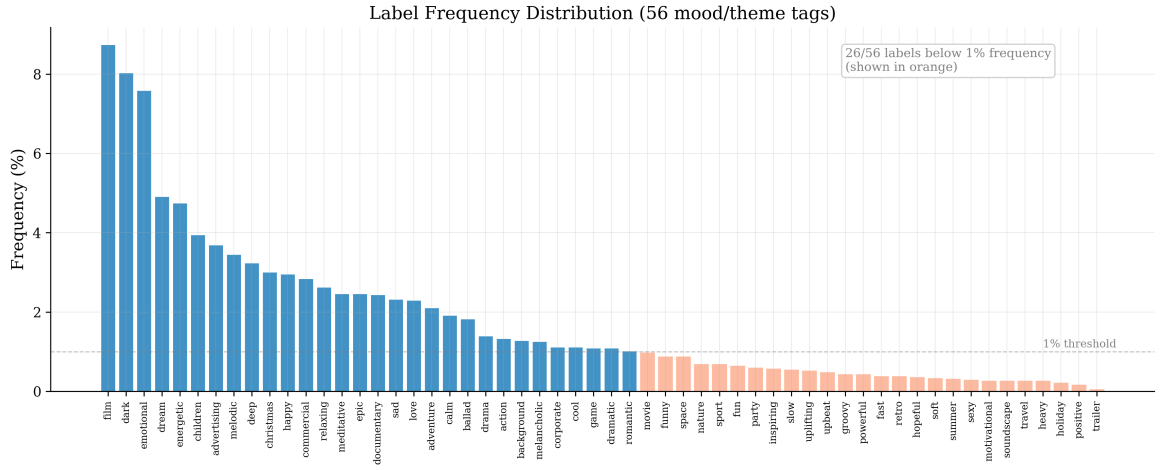

Figura 4.6: Distribución de frecuencias de las 56 etiquetas *mood/theme* en el conjunto de entrenamiento. Las barras en naranja corresponden a las 26 etiquetas con frecuencia inferior al 1 %.

De las 56 etiquetas, los modelos con modelado temporal (CRNN o CNN-Transformer) superan a la CNN en 34 etiquetas en ROC-AUC, mientras que la CNN supera a ambos modelos temporales en las 22 restantes. La frecuencia media de las etiquetas donde ganan los modelos temporales (1,48 %) es inferior a la de las etiquetas donde gana la CNN (2,26 %), lo que sugiere que el modelado temporal podría aportar más valor en emociones de baja frecuencia — aquellas que disponen de menos ejemplos de entrenamiento y donde la información sobre cómo evoluciona el audio a lo largo del tiempo podría compensar la escasez de datos.

La cuadro 4.6 presenta las 10 etiquetas con mayor ventaja del modelado temporal y la cuadro 4.7 las 10 con mayor ventaja de la CNN. La columna *n* indica el número de pistas positivas en test, Freq. la frecuencia relativa en entrenamiento, y Δ la diferencia de ROC-AUC entre el mejor modelo temporal y la CNN.

Tabla 4.6: 10 etiquetas donde los modelos temporales superan a la CNN. $\Delta = \max(\text{CRNN}, \text{Transformer}) - \text{CNN}$.

Etiqueta	*n*	**Freq.**	**CNN**	**CRNN**	**Trans.**	**Ganador**	Δ
holiday	9	0,21 %	0,459	0,648	0,705	Trans.	+0,246
retro	16	0,38 %	0,593	0,719	0,747	Trans.	+0,154
uplifting	22	0,52 %	0,585	0,678	0,712	Trans.	+0,127
background	54	1,28 %	0,648	0,684	0,731	Trans.	+0,082
soundscape	11	0,26 %	0,712	0,780	0,762	CRNN	+0,068
nature	29	0,69 %	0,514	0,582	0,564	CRNN	+0,068
motivational	11	0,26 %	0,661	0,720	0,630	CRNN	+0,059
relaxing	111	2,62 %	0,578	0,629	0,583	CRNN	+0,051
romantic	43	1,02 %	0,754	0,759	0,803	Trans.	+0,049
fast	16	0,38 %	0,870	0,898	0,914	Trans.	+0,044

Tabla 4.7: 10 etiquetas donde la CNN supera a los modelos temporales. $\Delta = \max(\text{CRNN}, \text{Transformer}) - \text{CNN}$. Los modelos temporales ganan en 34 de las 56 etiquetas y la CNN en las 22 restantes.

Etiqueta	*n*	**Freq.**	**CNN**	**CRNN**	**Trans.**	Δ
slow	23	0,54 %	0,658	0,520	0,553	−0,106
summer	13	0,31 %	0,716	0,625	0,659	−0,057
melodic	146	3,45 %	0,711	0,638	0,658	−0,052
positive	7	0,17 %	0,767	0,719	0,651	−0,048
christmas	127	3,00 %	0,761	0,714	0,712	−0,047
upbeat	20	0,47 %	0,766	0,695	0,723	−0,043
dark	340	8,04 %	0,754	0,664	0,714	−0,039
cool	47	1,11 %	0,710	0,659	0,674	−0,036
commercial	120	2,84 %	0,737	0,669	0,703	−0,034
game	46	1,09 %	0,735	0,702	0,677	−0,033

Las etiquetas donde el modelado temporal aporta mayor ventaja tienden a ser de muy baja frecuencia (7 de las 10 tienen frecuencia inferior al 1 %) y de carácter dinámico — categorías cuyas características sonoras podrían evolucionar a lo largo de la canción. Por ejemplo, una pista «uplifting» típicamente arranca contenida y va ganando energía, y un «soundscape» se construye mediante capas que aparecen y desaparecen. La CRNN y el Transformer pueden captar esa evolución porque analizan el audio como una secuencia ordenada en el tiempo, mientras que la CNN resume toda la canción en un único vector promedio y pierde esa información de «antes y después».

Las etiquetas donde la CNN presenta mayor ventaja tienden a ser de mayor frecuencia y de carácter más estable — categorías como «dark», «melodic» o «christmas» cuyas características sonoras (timbre oscuro, melodía predominante, instrumentación navideña) se mantienen relativamente constantes a lo largo de la canción. En estos casos la información temporal no aporta valor adicional y podría incluso confundir al modelo, haciéndole buscar patrones de evolución donde en realidad la señal relevante es estática.

La etiqueta con el peor rendimiento absoluto en todos los modelos es *movie* (CNN: 0,378; CRNN: 0,359; Transformer: 0,230), la única cuya ROC-AUC se sitúa consistentemente por debajo de 0,5 en todas las arquitecturas, lo que indica que la señal acústica no contiene información suficiente para distinguir esta categoría. Esto es esperable: «movie» no describe una emoción sino un contexto de uso, y no tiene un sonido propio reconocible — una banda sonora puede sonar a cualquier cosa. En el extremo opuesto, *deep* alcanza ROC-AUC de entre 0,910 y 0,944 según el modelo (CNN: 0,931; CRNN: 0,944; Transformer: 0,910), siendo la etiqueta más fácilmente reconocible, posiblemente porque se asocia a características sonoras muy definidas (frecuencias graves sostenidas, ritmo pausado).

Cabe señalar que estas interpretaciones se basan en un único *split* (*split-0*) y una única semilla (*seed* = 42). Las tendencias observadas son consistentes y plausibles, pero su generalización requeriría validación con múltiples particiones y semillas.

La figura 4.7 sintetiza visualmente todo el análisis anterior y constituye la representación central de este trabajo. La clave para leerla es fijarse en dos cosas: la posición horizontal de los puntos (cuanto más a la derecha, mejor reconoce el modelo esa emoción) y la separación entre los tres puntos de cada fila (cuanto más juntos están, más se parecen los tres modelos en esa etiqueta; cuanto más separados, mayor es la diferencia entre arquitecturas).

Cada fila es una de las 56 etiquetas, ordenadas de mayor dificultad arriba a menor abajo. La línea vertical punteada marca el azar (0,50): a su izquierda los modelos no distinguen esa emoción. Lo que la figura muestra es que en la gran mayoría de las etiquetas los tres puntos aparecen muy juntos — los modelos rinden de forma similar. Sin embargo, en ciertas filas los puntos se separan visiblemente: ahí es donde las diferencias de arquitectura importan. Por ejemplo, en «holiday» o «retro» (zona superior) los puntos temporales se desplazan claramente a la derecha del punto CNN, reflejando la ventaja del modelado temporal discutida en las tablas anteriores. En el extremo superior, «movie» es la única etiqueta donde todos los puntos quedan a la izquierda de la línea del azar: ningún modelo consigue reconocerla.

La conclusión que se extrae de esta figura es doble: (1) la arquitectura no es el factor determinante del rendimiento — la dificultad intrínseca de cada emoción pesa mucho más que la elección del modelo, y (2) las diferencias entre modelos, cuando existen, se concentran en etiquetas específicas y no se distribuyen uniformemente, lo que explica por qué el promedio global (ROC-AUC macro) no muestra diferencias significativas entre las tres arquitecturas.

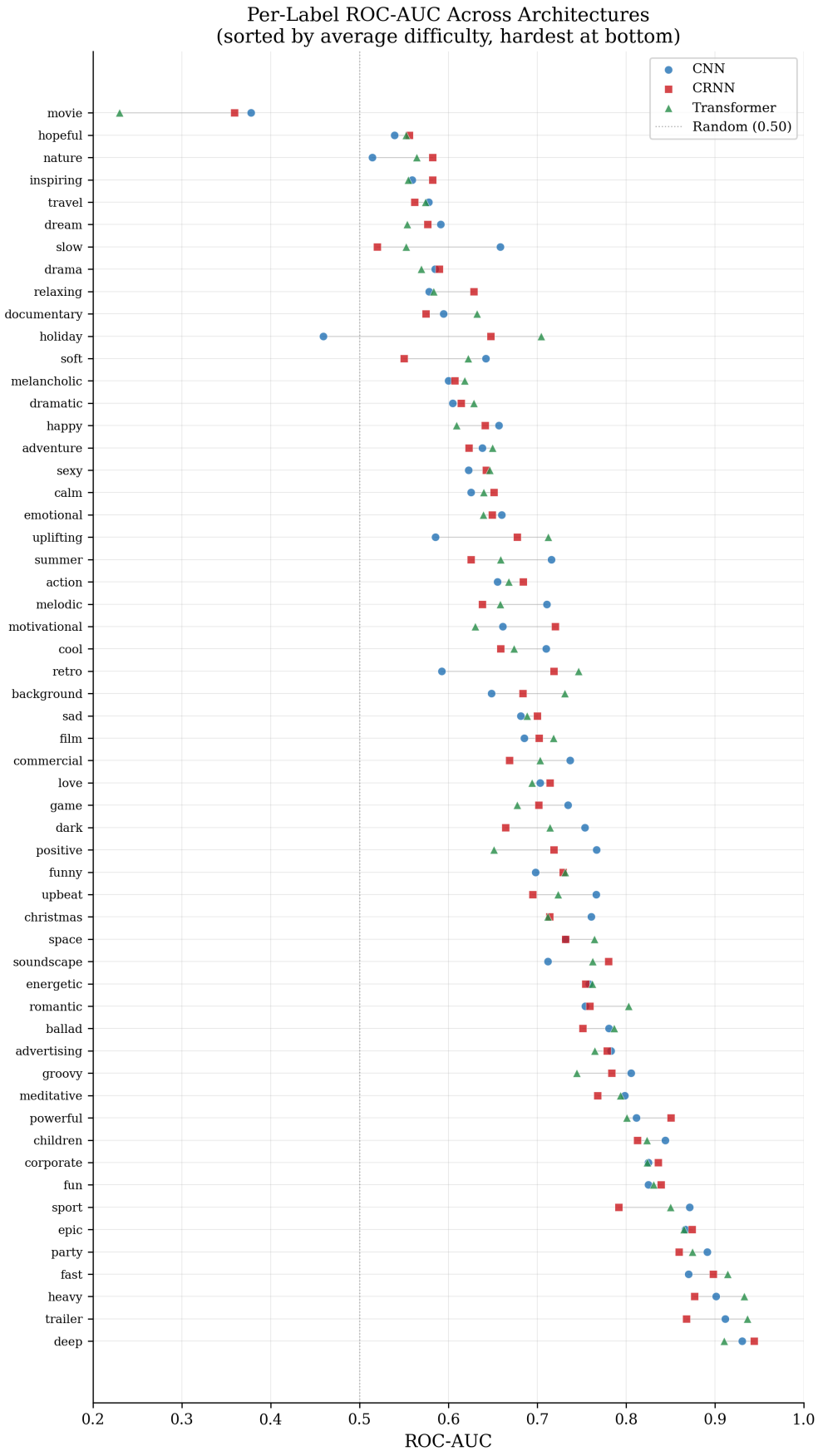

Figura 4.7: ROC-AUC por etiqueta para las tres arquitecturas. Cada fila es una etiqueta, ordenadas de mayor dificultad (arriba) a menor (abajo). La línea punteada vertical indica el azar (0,50). Puntos juntos indican rendimiento similar entre modelos; puntos separados señalan las etiquetas donde la elección de arquitectura marca diferencia.

4.6.4. Validación estadística y control experimental

Que un modelo obtenga una ROC-AUC de 0,7029 y otro 0,6988 no significa necesariamente que el primero sea mejor: la diferencia podría deberse simplemente a qué canciones concretas componen el conjunto de test. Para distinguir las diferencias reales de las que son fruto del azar, se aplicaron dos procedimientos estadísticos complementarios. Ambos se ejecutaron sobre un único *split* (*split-0*) con semilla fija (*seed* = 42), por lo que los resultados deben interpretarse como evidencia dentro de este marco experimental concreto.

Bootstrap (1.000 remuestreos)

El *bootstrap* es un método que permite estimar cuánto variaría una métrica si se dispusiera de diferentes conjuntos de test. En lugar de recoger nuevos datos (lo cual no es viable), se simulan nuevos conjuntos de test extrayendo muestras con reemplazo del conjunto existente, 1.000 veces. Para cada versión simulada se recalcula la ROC-AUC, obteniendo así una distribución de valores que indica el rango dentro del cual se sitúa la métrica con un 95 % de confianza (intervalo de confianza).

Tabla 4.8: Intervalos de confianza al 95 % para la ROC-AUC macro de cada modelo, obtenidos mediante *bootstrap* con 1.000 remuestreos del conjunto de test.

Modelo	ROC-AUC	IC 95 %	Ancho IC
CNN Baseline	0,7022	[0,6894 ; 0,7143]	0,0249
CRNN	0,6988	[0,6870 ; 0,7104]	0,0234
CNN-Transformer	0,7029	[0,6906 ; 0,7135]	0,0229

Las comparaciones por pares confirman que ninguna diferencia alcanza significación estadística:

Tabla 4.9: Comparaciones por pares de ROC-AUC macro. La proporción empírico se calcula como la proporción de remuestreos *bootstrap* en los que la diferencia observada cambia de signo. Valores de p-valor superiores a 0,05 se consideran no significativos.

Comparación	Δ AUC	IC 95 %	Proporción
CRNN − CNN	−0,0034	[−0,012 ; +0,005]	0,400
Transformer − CNN	+0,0007	[−0,008 ; +0,010]	0,890
Transformer − CRNN	+0,0041	[−0,004 ; +0,012]	0,342

Los tres intervalos de confianza se solapan ampliamente y todos los p-valores superan 0,34. A nivel macro, las tres arquitecturas son estadísticamente indistinguibles con la evidencia disponible.

Test de DeLong por etiqueta

El *bootstrap* anterior responde a la pregunta «¿son los modelos diferentes a nivel global?». Pero como se ha visto en la apartado 4.6.3, las diferencias se cancelan al promediar. Para detectar en qué etiquetas concretas un modelo es significativamente mejor que otro, se aplicó el test de DeLong [32] — un test estadístico diseñado específicamente para comparar dos curvas ROC— etiqueta a etiqueta (55 etiquetas testables, excluyendo *trailer* con solo 2 positivos en test). Los resultados ($p < 0{,}05$) muestran diferencias significativas en un número sustancial de etiquetas:

- CRNN vs CNN: 15 de 55 etiquetas con diferencia significativa.
- Transformer vs CNN: 19 de 55 etiquetas con diferencia significativa.
- Transformer vs CRNN: 12 de 55 etiquetas con diferencia significativa.

Conviene señalar que estos resultados no incluyen corrección por comparaciones múltiples (como Bonferroni o FDR). Al realizar 55 tests simultáneos por cada par de modelos, es esperable que algunas diferencias resulten significativas por azar (aproximadamente 2–3 de cada 55 al nivel $p < 0{,}05$). No obstante, el número de etiquetas significativas observado (12–19) supera ampliamente ese umbral esperado, lo que sugiere que una proporción sustancial de las diferencias son reales.

Esto es compatible con el hallazgo central del análisis por etiqueta: las diferencias entre arquitecturas existen a nivel individual pero se cancelan al agregarse, porque los modelos temporales mejoran en unas etiquetas y empeoran en otras con respecto a la CNN, produciendo un empate aparente en las métricas macro. El mecanismo de agregación temporal no parece producir una mejora uniforme, sino una redistribución del rendimiento entre etiquetas.

4.6.5. Verificación del sistema

En un proyecto de *Deep Learning*, un error silencioso en el *pipeline* de datos o en la arquitectura del modelo puede invalidar todos los resultados experimentales sin producir ningún mensaje de error visible. Por ejemplo, si las particiones de entrenamiento y test comparten muestras por error, las métricas estarán infladas artificialmente; si una capa del modelo no recibe gradientes durante el entrenamiento, esa parte de la red no aprende nada aunque el entrenamiento parezca funcionar. Para detectar este tipo de problemas antes de lanzar los entrenamientos finales — que requieren horas de cómputo—, se implementó un conjunto de tests automatizados que verifican la corrección del sistema. La cuadro 4.10 resume el plan de pruebas.

Tabla 4.10: Plan de pruebas del sistema. Cada test se implementó como un *script* independiente ejecutable antes del entrenamiento final.

Test	Objetivo	Criterio de éxito
Configuración	Comprobar que los parámetros del proyecto se cargan correctamente	Rutas existen, constantes coinciden con los valores esperados, semilla reproducible
Particiones (*splits*)	Verificar la carga de las particiones oficiales y la ausencia de *data leakage*	Cero solapamiento entre train/val/test; conteos coinciden con los del *dataset*
Etiquetas (*labels*)	Validar la codificación *multi-hot* de las 56 etiquetas *mood/theme*	56 etiquetas únicas; vectores binarios con forma $(56,)$; toda pista tiene al menos una etiqueta
Preprocesado	Comprobar que el `Dataset` de PyTorch devuelve muestras con la forma y tipo correctos	Espectrogramas con forma $(1, 96, 1366)$, `dtype float32`; etiquetas binarias
`DataLoaders`	Verificar la construcción de lotes y el comportamiento de aleatorización	Lotes con dimensiones correctas; train aleatorio, val/test deterministas
Arquitectura CNN	Validar *forward/backward pass* y flujo de gradientes de la CNN	Salida (batch, 56); gradientes en todas las capas entrenables
Integración CRNN	Validar *forward/backward* con datos reales y *batches* de tamaño variable	Funciona con *batches* de tamaño variable; gradientes correctos con datos reales
Sobreajuste en miniatura	Confirmar que el ciclo completo de entrenamiento funciona de extremo a extremo	Pérdida se reduce más del 50 % en 20 épocas sobre 100 muestras; sin NaN/Inf

Los tests se organizan en cuatro niveles de verificación progresivos:

1. **Integridad de los datos** (configuración, particiones, etiquetas). Comprueban que los datos llegan al modelo tal como se espera. Si las particiones se solaparan, el modelo podría «hacer trampa» evaluándose sobre datos ya vistos (*data leakage*); si las etiquetas estuvieran mal codificadas, el modelo optimizaría un objetivo incorrecto.

2. **Corrección dimensional** (preprocesado, `DataLoaders`). Verifican que cada etapa produce tensores con las dimensiones esperadas. Un desajuste dimensional no siempre provoca un error explícito: PyTorch puede aplicar *broadcasting* automático y producir resultados numéricos incorrectos sin avisar.

3. **Flujo de gradientes** (arquitectura CNN, integración CRNN). Confirman que el *forward* y *backward pass* funcionan correctamente y que los gradientes llegan a todas las capas entrenables. Si una capa estuviera desconectada, esa parte del modelo no aprendería.

4. **Capacidad de aprendizaje** (sobreajuste en miniatura). La prueba final: si un modelo no es capaz de memorizar 100 muestras en 20 épocas, algo está fundamentalmente mal en la implementación.

La ejecución satisfactoria de todas estas pruebas previa al entrenamiento final aporta confianza en que los resultados experimentales no están afectados por errores en la implementación del *pipeline*.

4.6.6. Cambios respecto a los objetivos iniciales

El desarrollo se ha ajustado en lo esencial a los objetivos definidos en el capítulo 3. Se identifican dos desviaciones menores respecto a la configuración prevista inicialmente, ambas derivadas de restricciones prácticas encontradas durante la experimentación:

1. **Tamaño de *batch*.** La configuración inicial preveía un tamaño de *batch* de 32 muestras. Sin embargo, con los modelos de mayor tamaño (CRNN, CNN-Transformer), el entrenamiento con *batch* = 32 producía errores de memoria en el dispositivo utilizado (Apple Silicon con *backend* MPS de PyTorch). Por ello, se redujo el tamaño de *batch* a 16 de forma consistente en los tres modelos, manteniendo así la comparabilidad entre arquitecturas. Este cambio podría tener un efecto menor sobre la convergencia (*batches* más pequeños introducen más ruido en la estimación del gradiente), pero no se espera que altere las conclusiones comparativas, ya que la modificación se aplicó por igual a los tres modelos.

2. **Tasa de aprendizaje del CNN-Transformer.** Con lr = 1e-3 el Transformer divergió: ROC-AUC de validación cayó de 0,6215 a 0,4890 en seis épocas, mientras que la pérdida de entrenamiento descendía normalmente. Esto indica un fallo de optimización: la tasa de aprendizaje elevada destruía los patrones de atención tras la primera época. Este comportamiento es consistente con la sensibilidad documentada de los Transformers a la tasa de aprendizaje [2, 21]. Se re-entrenó con lr = 1e-4 (la tasa estándar en la literatura de MIR), sin búsqueda de hiperparámetros sobre los datos. Esta diferencia podría favorecer o perjudicar al Transformer — no es posible determinar la dirección del efecto sin validación adicional.

No se han detectado otras desviaciones respecto al plan experimental original.

Capítulo 5

Discusión

Este capítulo interpreta los resultados del capítulo 4 sin repetirlos. El objetivo es responder a la pregunta central del trabajo —¿aporta el modelado temporal una ventaja real en el reconocimiento de emociones musicales?— y contextualizar la respuesta frente a la literatura, las limitaciones del protocolo y las implicaciones prácticas.

5.1. Interpretación de los resultados principales

Los tres modelos convergen en una ROC-AUC macro de test en torno a 0,70 (apartado 4.6), con una diferencia máxima de 0,41 puntos porcentuales y sin significación estadística en ninguna comparación por pares (p-valores $>$ 0,34, apartado 4.6.4). Esta convergencia sugiere que, bajo este protocolo, el rendimiento está fuertemente condicionado por la dificultad de la tarea más que por la elección de arquitectura: el subconjunto *mood/theme* combina desbalance extremo (26 de 56 etiquetas con frecuencia inferior al 1 %) con ambigüedad semántica, al mezclar descriptores afectivos (*happy*, *sad*) con etiquetas de contexto de uso (*movie*, *corporate*) que carecen de un correlato acústico estable [1, 8].

Los extremos del espectro lo ilustran con claridad: *movie* es la única etiqueta con ROC-AUC inferior a 0,5 en los tres modelos — describe para qué se usa una pieza, no cómo suena —, mientras que *deep* alcanza 0,91–0,94, porque sus características acústicas (graves sostenidos, tempo pausado) son consistentes y reconocibles (apartado 4.6.3).

En comparación con el *baseline* reportado en MediaEval 2019 para *mood/theme* (ROC-AUC $\approx$ 0,725, con un protocolo intensivo y umbrales por etiqueta) [1], los resultados quedan 2–3 pp por debajo. Esta diferencia es esperable: los modelos de este trabajo se entrenan desde cero, sin *transfer learning*, sobre hardware de consumo. Que modelos sin pre-entrenamiento se sitúen tan cerca de esa referencia refleja la solidez del pipeline experimental, no una limitación.

5.2. El modelado temporal: redistribución, no mejora uniforme

La pregunta central recibe una respuesta matizada: el modelado temporal no produce una mejora general a nivel macro, pero tampoco es indiferente. El análisis por etiqueta (apartado 4.6.3) revela que los modelos temporales superan a la CNN en 34 de las 56 etiquetas, mientras que la CNN gana en las 22 restantes. El resultado neto es un empate aparente. Este hallazgo es uno de los más relevantes del trabajo: las métricas agregadas no muestran diferencias significativas no porque la arquitectura sea irrelevante, sino porque sus efectos se cancelan al promediar.

El patrón subyacente es coherente: las etiquetas donde los modelos temporales destacan (*holiday*, *retro*, *uplifting*, *background*) son de baja frecuencia y carácter dinámico — emociones cuya señal acústica evoluciona a lo largo de la canción. La LSTM y el mecanismo de atención capturan ese arco temporal; la CNN, al colapsar la dimensión temporal mediante *pooling* global, lo pierde. Las etiquetas donde la CNN domina (*dark*, *melodic*, *christmas*) presentan el perfil opuesto: alta frecuencia y características sonoras estables, donde la información temporal es redundante e incluso contraproducente.

El test de DeLong (apartado 4.6.4) apoya que parte de estas diferencias no se debe únicamente al azar: entre 12 y 19 etiquetas por par de modelos muestran diferencias significativas ($p < 0{,}05$), un número que supera ampliamente las 2–3 esperables por azar sin corrección por comparaciones múltiples.

Otro hallazgo notable es el comportamiento de la CRNN: es el único modelo con transferencia negativa de validación a test ($-1{,}04$ pp). Obtiene la mejor ROC-AUC en validación (0,7092) pero la peor en test (0,6988), lo que es consistente con la hipótesis de que la LSTM podría estar sobreajustando patrones temporales específicos de los artistas de validación que no generalizan a artistas no vistos, dada la partición por artista del MTG-Jamendo.

Finalmente, la optimización de umbrales (apartado 4.6) muestra que el F1 macro mejora entre 7 y 16 veces con umbrales por etiqueta calibrados sobre validación. Este resultado evidencia que el umbral fijo $t = 0{,}5$ es inadecuado bajo desbalance extremo, independientemente de la arquitectura, y que la calibración por etiqueta es un paso clave para que un sistema sea operativo en escenarios con desbalance extremo como el de este *benchmark*.

5.3. Limitaciones del estudio

Cuatro limitaciones condicionan el alcance de las conclusiones:

1. **Único *split* y semilla (*split-0*, seed = 42).** No es posible estimar la varianza debida a la inicialización aleatoria o a la composición de la partición de test. El *bootstrap* de la apartado 4.6.4 mitiga parcialmente el problema, pero no sustituye a la validación cruzada completa.

2. **Tasa de aprendizaje asimétrica del CNN-Transformer** (lr = 10^{-4} frente a 10^{-3} para CNN y CRNN, apartado 4.6.6). Era necesaria — el Transformer diverge con lr = 10^{-3}, un comportamiento documentado [21] —, pero introduce una variable no controlada cuya dirección de efecto es indeterminada.

3. **Reducción de batch de 32 a 16** (apartado 4.6.6). Simétrica entre modelos, pero *batches* más pequeños introducen más ruido en la estimación del gradiente.

4. **Ausencia de pre-entrenamiento y aumentación de datos.** Si el *transfer learning* beneficiara de forma diferencial a alguna arquitectura, las conclusiones comparativas podrían cambiar [2].

5.4. Implicaciones del estudio

Los resultados tienen cuatro implicaciones concretas:

1. **Recomendación práctica.** Dado que los tres modelos son estadísticamente indistinguibles a nivel macro, bajo este protocolo la CNN Baseline (1,58M parámetros) es la opción más eficiente: comparable en ROC-AUC, superior en mAP (0,0652 frente a 0,0639 y 0,0595), y con el menor coste computacional. En entornos de producción o recursos limitados, añadir complejidad arquitectónica no se traduce en beneficio global.

2. **Las métricas macro ocultan diferencias reales.** El análisis por etiqueta demuestra que comparar modelos solo mediante ROC-AUC macro puede llevar a la conclusión incorrecta de que la arquitectura es irrelevante, cuando en realidad produce efectos focalizados que un sistema de producción debería considerar según qué emociones son críticas para su aplicación. Este hallazgo es relevante para el diseño de futuros benchmarks en MIR.

3. **El resultado negativo es informativo.** La ausencia de diferencia macro significativa es consistente con lo observado por Won et al. [2]: cuando se controlan las condiciones experimentales, las mejoras atribuibles a la arquitectura tienden a reducirse. En este *benchmark*, el desbalance y la heterogeneidad semántica de *mood/theme* pueden limitar la ganancia agregada. Este trabajo ayuda a acotar, bajo un protocolo controlado, qué cambios arquitectónicos producen redistribuciones por etiqueta sin mejora macro.

4. **La calibración de umbrales es imprescindible.** Sin ella, 54 de 56 etiquetas producen F1 = 0; con umbrales optimizados, el número baja a 9–14. Esto no es una particularidad de estos modelos sino una consecuencia del desbalance extremo, un fenómeno ubicuo en bases de datos de audio con anotación libre. Cualquier sistema operativo de clasificación de emociones musicales necesita este paso de calibración.

Capítulo 6

Conclusiones

6.1. Conclusiones del trabajo

Este trabajo responde una pregunta central: ¿aporta el modelado temporal una ventaja real en reconocimiento de emociones musicales? La respuesta no es binaria, sino un descubrimiento sobre cómo opera esa ventaja.

A nivel global, los tres modelos convergen en ROC-AUC macro ~ 0,70 con diferencias no significativas. Esta aparente equivalencia podría sugerir irrelevancia arquitectónica. Sin embargo, el análisis por etiqueta revela el mecanismo real: los modelos temporales superan a la CNN en 34 de 56 emociones y la CNN en las 22 restantes. Es una redistribución pura — mejora en unas que se compensa con retroceso en otras—, no una mejora uniforme. Las métricas agregadas ocultan este efecto; solo el análisis por etiqueta con validación estadística (test de DeLong) lo detecta.

El patrón es coherente: emociones dinámicas de baja frecuencia (*holiday*, *retro*, *uplifting*) se benefician del modelado temporal; emociones estables de alta frecuencia (*dark*, *melodic*, *christmas*) se perjudican. Bajo este protocolo, los resultados sugieren que el rendimiento global está más condicionado por los datos que por la elección de arquitectura: desbalance extremo (26 de 56 etiquetas inferior al 1 ‰), ambigüedad semántica (*movie* describe un contexto de uso y, en este *benchmark*, no muestra un correlato acústico estable) y subjetividad anotativa. Este resultado es consistente con las observaciones de Won et al. [2] sobre la dificultad de atribuir mejoras de rendimiento a la arquitectura cuando se controlan estrictamente las condiciones experimentales. En este *benchmark*, el desbalance y la heterogeneidad semántica de *mood/theme* pueden limitar la ganancia agregada, incluso cuando se incorporan mecanismos temporales explícitos.

Los resultados del *pipeline* (ROC-AUC ~0,70) son sólidos bajo estas limitaciones. En comparación con el *baseline* reportado en MediaEval 2019 para *mood/theme* (ROC-AUC ~0,725, con un protocolo intensivo y umbrales por etiqueta) [1], la diferencia observada (2–3 pp) es esperable dado que aquí se entrena desde cero y sin técnicas adicionales. Un hallazgo práctico complementario: el umbral estándar $t = 0{,}5$ produce F1 = 0 en 54 de 56 etiquetas; la calibración por etiqueta lo multiplica entre 7 y 16 veces. Esta calibración es un paso clave en escenarios con desbalance extremo como el de este *benchmark*, y no depende de una arquitectura concreta.

La aportación principal es el descubrimiento del mecanismo redistributivo que las métricas globales enmascaran. Esto tiene implicación directa: el diseño de sistemas de reconocimiento emocional debe orientarse por las emociones críticas de cada aplicación, no por *benchmarks* convencionales que promedian efectos opuestos. Bajo protocolo experimental riguroso, la arquitectura sí importa, pero de formas localizadas que un análisis global nunca revelaría.

6.2. Conclusiones personales

La música ha sido parte central de mi vida desde siempre. En los últimos años descubrí el big data y el machine learning, dos mundos que me apasionaron por su capacidad de extraer significado de lo complejo. Cuando tuve la oportunidad de aplicar deep learning al reconocimiento de emociones musicales, no pude resistirme: era fusionar dos pasiones.

Lo que más he valorado de este trabajo no ha sido simplemente implementar modelos, sino aprender a pensar como un científico de datos. Específicamente, descubrir — a través del análisis minucioso por etiqueta — que las conclusiones globales a menudo ocultan realidades más complejas. Esto es una lección que va más allá de este dataset: en cualquier tarea con datos reales siempre hay matices. Una métrica agregada nunca cuenta la historia completa. Importa el contexto — en este caso, la naturaleza de cada emoción, su frecuencia, su ambigüedad semántica.

También he aprendido la importancia de la humildad metodológica. Al comenzar este proyecto daba por supuesto que una arquitectura más compleja — con memoria recurrente o atención — rendiría necesariamente mejor que una CNN simple. Era una intuición razonable, pero resultó incompleta. Los resultados globales mostraban empate, y eso al principio desconcertaba. Fue el análisis por etiqueta el que reveló lo que realmente estaba ocurriendo: la mejora existía, pero en emociones concretas, y se cancelaba al promediar. Esa experiencia me enseñó que en ciencia de datos las respuestas más interesantes raramente son las obvias, y que a menudo se esconden debajo de un promedio que las aplana.

Este proyecto me dejó una convicción personal: el machine learning es una herramienta extraordinaria, pero solo si la aplicamos con rigor y sin conformarnos con el primer número que obtenemos.

Capítulo 7

Futuras líneas de trabajo

Los resultados y limitaciones discutidos en los capítulos anteriores abren varias direcciones naturales para extender este trabajo. Se destacan las cuatro con mayor potencial de impacto.

- **Validación cruzada con múltiples *splits* y semillas.** La limitación más relevante de este estudio es el uso de un único *split* y una única semilla. El MTG-Jamendo dispone de cinco particiones oficiales por artista [7]. Repetir el experimento sobre las cinco, con varias semillas de inicialización, permitiría estimar la varianza real de las métricas y confirmar si el patrón de redistribución por etiqueta observado es estable o depende de la composición concreta de la partición. Esta es la extensión más directa y la que mayor solidez estadística aportaría a las conclusiones.

- **Transfer learning con modelos de audio pre-entrenados.** Los modelos de este trabajo aprenden desde cero. Recientemente, modelos como MERT [25] han demostrado que el pre-entrenamiento auto-supervisado a gran escala sobre audio musical mejora el rendimiento en múltiples tareas de MIR, incluyendo *music tagging*. Una línea de gran interés sería sustituir el *backbone* convolucional por *embeddings* pre-entrenados y evaluar si el Transformer — que en este estudio no muestra ventaja global — se beneficia diferencialmente del *transfer learning*, y analizar si el Transformer podría beneficiarse diferencialmente del *transfer learning* bajo este *benchmark*.

- **Reconocimiento multimodal: audio y letras.** Este trabajo opera exclusivamente sobre la señal de audio. Sin embargo, parte de la información emocional de una canción reside en sus letras. Una posible extensión es integrar audio y letras mediante un enfoque multimodal, ya que parte de la información emocional puede estar presente en el contenido textual. Este planteamiento podría ser especialmente relevante para etiquetas con mayor carga semántica o de contexto de uso, que en este estudio mostraron bajo rendimiento (*movie*, *corporate*). Incorporar la modalidad textual podría elevar el techo de rendimiento que los modelos únicamente acústicos no logran superar.

- **Aumentación de datos y estrategias ante desbalance.** Técnicas como SpecAugment, *mixup* o sobremuestreo de etiquetas minoritarias son candidatas para explorar su efecto sobre el rendimiento en las 26 etiquetas con frecuencia inferior al 1 %, que coinciden con aquellas donde el modelado temporal mostró mayor ventaja. Evaluar si la aumentación reduce la brecha entre arquitecturas o la amplifica sería un resultado de interés tanto práctico como científico.

Bibliografía

[1] D. Bogdanov, A. Porter, P. Tovstogan, and M. Won, "MediaEval 2019: Emotion and theme recognition in music using Jamendo," in *Proc. MediaEval 2019 Workshop*, 2019.

[2] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of CNN-based automatic music tagging models," in *Proc. 17th Sound and Music Computing Conference (SMC)*, 2020.

[3] Y.-H. Yang and H. H. Chen, "Machine recognition of music emotion: A review," *ACM Transactions on Intelligent Systems and Technology*, vol. 3, no. 3, pp. 40:1–40:30, 2012.

[4] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[5] R. E. Thayer, *The Biopsychology of Mood and Arousal*. Oxford University Press, 1989.

[6] T. Eerola and J. K. Vuoskoski, "A comparison of the discrete and dimensional models of emotion in music," *Psychology of Music*, vol. 39, no. 1, pp. 18–49, 2011.

[7] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The MTG-Jamendo dataset for automatic music tagging," in *Machine Learning for Music Discovery Workshop, ICML*, 2019.

[8] R. Delbouys, R. Hennequin, F. Piccoli, J. Royo-Letelier, and M. Moussallam, "Music mood detection based on audio and lyrics with deep neural net," in *Proc. 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[9] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," in *International Journal of Data Warehousing and Mining*, vol. 3, 2010, pp. 1–13.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[11] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proc. 17th International Society for Music Information Retrieval Conference*

(ISMIR), 2016.

[12] M. Müller, *Fundamentals of Music Processing*. Springer, 2015.

[13] J. B. Allen and L. R. Rabiner, "A unified approach to short-time fourier analysis and synthesis," in *Proc. IEEE*, vol. 65, no. 11, 1977, pp. 1558–1564.

[14] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[15] J. Lee, J. Park, K. L. Kim, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," *arXiv preprint arXiv:1703.01789*, 2017.

[16] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[22] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.

[23] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio spectrogram transformer," in *Proc. Interspeech*, 2021.

[24] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: A hierarchical

token-semantic audio transformer for sound classification and detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[25] Y. Li, R. Yuan, G. Zhang *et al.*, "MERT: Acoustic music understanding model with large-scale self-supervised training," in *International Conference on Learning Representations (ICLR)*, 2024.

[26] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[27] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proc. 23rd International Conference on Machine Learning (ICML)*, 2006, pp. 233–240.

[28] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proc. 10th International Society for Music Information Retrieval Conference (ISMIR)*, 2009.

[29] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[31] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proc. 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[32] X. Sun and W. Xu, "Fast implementation of DeLong's algorithm for comparing the areas under correlated receiver operating characteristic curves," *IEEE Signal Processing Letters*, vol. 21, no. 11, pp. 1389–1393, 2014.

Apéndice A

Configuración experimental y reproducibilidad

Este anexo documenta la configuración utilizada en los experimentos del presente trabajo con el objetivo de facilitar su reproducibilidad y auditar las condiciones del protocolo experimental.

A.1. Hiperparámetros de entrenamiento

La cuadro A.1 recoge los hiperparámetros efectivos empleados en el entrenamiento de los tres modelos. El valor marcado con un asterisco (*) es asimétrico entre modelos y se discute como limitación en la apartado 5.3.

Tabla A.1: Hiperparámetros de entrenamiento por modelo.

Parámetro	**CNN**	**CRNN**	**Transformer**
Optimizador	Adam	Adam	Adam
Tasa de aprendizaje	10^{-3}	10^{-3}	10^{-4} *
Tamaño de lote	16	16	16
Épocas máximas	100	100	100
Early stopping (paciencia)	5	5	5
Mejor época (de total)	19 / 24	25 / 30	10 / 15
Función de pérdida	`BCEWithLogitsLoss`		
Tasa de *dropout*	0.5	0.5	0.5
Semilla global	42		

* El Transformer diverge con lr = 10^{-3}; se redujo a 10^{-4} tras verificación experimental (véase apartado 4.6.6). Este es el único hiperparámetro asimétrico entre modelos.

A.2. Pipeline de datos

Tabla A.2: Especificación del *pipeline* de datos.

Parámetro	Valor
Dataset	MTG-Jamendo `autotagging_moodtheme`
Número de etiquetas	56 (*mood/theme*)
Partición	*split-0* (oficial por artista)
Representación de entrada	Mel-espectrogramas precomputados (`.npy`)
Bandas mel (`N_MELS`)	96
Ventana temporal (`N_FRAMES`)	1 366 *frames*
Crop/pad	*center crop* (si $T > 1366$) / *zero-pad* centrado (si $T < 1366$)
Normalización	No se aplica normalización adicional en este trabajo
Codificación de etiquetas	*multi-hot* (vector de 56 posiciones, $\{0, 1\}$)
Aumentación de datos	Ninguna

A.3. Entorno de software y hardware

Tabla A.3: Entorno de ejecución.

Componente	Especificación
Python	3.10
Dependencias	Documentadas en `requirements.txt`
Sistema operativo	macOS
Hardware	Portátil personal (Apple Silicon), 16 GB RAM
Dispositivo de cálculo	Apple MPS (Metal Performance Shaders)

A.4. Estrategia de reproducibilidad

Para reducir la variabilidad debida a la inicialización aleatoria, se fija una semilla global al inicio de cada entrenamiento. Se aplica:

- `random.seed(42)`
- `numpy.random.seed(42)`
- `torch.manual_seed(42)`

En ejecución sobre GPU NVIDIA (CUDA), adicionalmente se fijan opciones de cuDNN para favorecer determinismo (cuando aplica). En MPS (Apple Silicon), la fijación de semilla reduce la variabilidad, pero no garantiza reproducibilidad bit-a-bit.

Los tres modelos comparten el mismo *backbone* CNN y siguen el mismo protocolo experimental. Esto permite atribuir las diferencias de rendimiento principalmente al mecanismo de agregación temporal, dentro de las limitaciones discutidas en la apartado 5.3.

Apéndice B

Resultados completos por etiqueta y significancia estadística

Este anexo presenta los resultados completos del conjunto de test para las 56 etiquetas *mood/theme*, así como un resumen de los tests de significancia estadística (DeLong) por etiqueta. Las tablas y figuras del cuerpo de la memoria muestran extractos representativos; el presente anexo permite la verificación exhaustiva.

B.1. ROC-AUC por etiqueta

La cuadro B.1 recoge, para cada una de las 56 etiquetas, la frecuencia relativa en el conjunto de entrenamiento, el número de muestras positivas en test (n_+), y el ROC-AUC alcanzado por cada modelo. La columna *Mejor* indica el modelo con el ROC-AUC más alto. Las etiquetas están ordenadas alfabéticamente.

Tabla B.1: ROC-AUC en el conjunto de test para las 56 etiquetas *mood/theme*. Valores en **negrita** indican el mejor modelo por etiqueta (en caso de empate numérico, se mantiene el primero por orden en la tabla).

Etiqueta	Frec. (%)	n_+	CNN	CRNN	Trans.	Mejor
action	1.32	56	.655	**.684**	.668	CRNN
adventure	2.10	89	.638	.623	**.650**	Trans.
advertising	3.69	156	**.783**	.779	.765	CNN
background	1.28	54	.648	.684	**.731**	Trans.
ballad	1.82	77	.781	.751	**.787**	Trans.
calm	1.91	81	.626	**.651**	.640	CRNN
children	3.95	167	**.844**	.813	.823	CNN
christmas	3.00	127	**.761**	.714	.712	CNN
commercial	2.84	120	**.737**	.669	.703	CNN
cool	1.11	47	**.710**	.659	.674	CNN
corporate	1.11	47	.825	**.836**	.824	CRNN
dark	8.04	340	**.754**	.664	.714	CNN

Continúa en la siguiente página

Etiqueta	Frec. (%)	n_+	CNN	CRNN	Trans.	Mejor
deep	3.24	137	.931	**.944**	.910	CRNN
documentary	2.43	103	.594	.575	**.632**	Trans.
drama	1.39	59	.585	**.590**	.570	CRNN
dramatic	1.09	46	.605	.614	**.629**	Trans.
dream	4.92	208	**.591**	.577	.554	CNN
emotional	7.59	321	**.660**	.649	.639	CNN
energetic	4.75	201	.758	.754	**.762**	Trans.
epic	2.46	104	.867	**.874**	.865	CRNN
fast	0.38	16	.870	.898	**.914**	Trans.
film	8.75	370	.685	.702	**.718**	Trans.
fun	0.64	27	.825	**.839**	.831	CRNN
funny	0.87	37	.698	.729	**.732**	Trans.
game	1.09	46	**.735**	.702	.677	CNN
groovy	0.43	18	**.806**	.784	.744	CNN
happy	2.95	125	**.657**	.641	.609	CNN
heavy	0.26	11	.901	.877	**.933**	Trans.
holiday	0.21	9	.459	.648	**.705**	Trans.
hopeful	0.35	15	.540	**.556**	.552	CRNN
inspiring	0.57	24	.559	**.582**	.555	CRNN
love	2.29	97	.703	**.714**	.694	CRNN
meditative	2.46	104	**.798**	.768	.794	CNN
melancholic	1.25	53	.600	.607	**.619**	Trans.
melodic	3.45	146	**.711**	.638	.658	CNN
motivational	0.26	11	.661	**.720**	.630	CRNN
movie	0.97	41	**.378**	.359	.230	CNN
nature	0.69	29	.514	**.582**	.564	CRNN
party	0.59	25	**.891**	.859	.875	CNN
positive	0.17	7	**.767**	.719	.651	CNN
powerful	0.43	18	.812	**.850**	.801	CRNN
relaxing	2.62	111	.578	**.629**	.583	CRNN
retro	0.38	16	.593	.719	**.747**	Trans.
romantic	1.02	43	.754	.759	**.803**	Trans.
sad	2.32	98	.681	**.700**	.688	CRNN
sexy	0.28	12	.623	.643	**.647**	Trans.
slow	0.54	23	**.658**	.520	.553	CNN
soft	0.33	14	**.642**	.550	.622	CNN
soundscape	0.26	11	.712	**.780**	.762	CRNN
space	0.87	37	.732	.732	**.764**	Trans.
sport	0.69	29	**.872**	.792	.850	CNN
summer	0.31	13	**.716**	.625	.659	CNN
trailer	0.05	2	.912	.868	**.937**	Trans.
travel	0.26	11	**.578**	.562	.574	CNN
upbeat	0.47	20	**.766**	.695	.723	CNN
uplifting	0.52	22	.585	.677	**.712**	Trans.

Resumen de conteo por modelo con mejor ROC-AUC: CNN gana en 22 de 56 etiquetas, CRNN en 15 y el Transformer en 19. Estas ventajas se compensan al promediar, resultando en métricas macro estadísticamente equivalentes.

B.2. Test de DeLong por etiqueta

La cuadro B.2 resume las etiquetas donde el test de DeLong [32] detecta diferencias significativas ($p < 0{,}05$, sin corrección por comparaciones múltiples) en cada comparación por pares. Se omiten las etiquetas no significativas por brevedad; los resultados completos se generan a partir de `statistical_analysis_results.json`. La etiqueta *trailer* ($n_+ = 2$) fue excluida del test por muestra insuficiente.

Las tablas anteriores evidencian la naturaleza bidireccional de las diferencias: no existe un modelo uniformemente superior. Este patrón de redistribución se cancela al calcular promedios macro, lo que explica la equivalencia estadística reportada en el cuerpo de la memoria.

Tabla B.2: Etiquetas con diferencia de ROC-AUC significativa (DeLong, $p < 0{,}05$) en cada comparación por pares. Se reportan z y p; la dirección de la diferencia se interpreta comparando los valores de ROC-AUC mostrados. Etiquetas también significativas a $p < 0{,}01$ marcadas con †.

(a) CRNN vs. CNN (15 / 55 etiquetas significativas; 7 a $p < 0{,}01$)

Etiqueta	AUC_{CNN}	AUC_{CRNN}	z	p
children†	.844	.813	+3.18	.0015
christmas	.761	.714	+2.14	.033
commercial†	.737	.669	+4.08	4.4 e−5
dark†	.754	.664	+7.77	8.2 e−15
deep	.931	.944	−2.16	.031
holiday	.459	.648	−2.23	.026
meditative	.798	.768	+1.97	.049
melodic†	.711	.638	+3.32	.0009
party	.891	.859	+2.19	.029
powerful	.812	.850	−2.02	.044
relaxing†	.578	.629	−3.51	.0004
slow†	.658	.520	+3.34	.0008
sport	.872	.792	+2.52	.012
summer	.716	.625	+2.17	.030
upbeat†	.766	.695	+3.26	.001

(b) Transformer vs. CNN (19 / 55 etiquetas significativas; 8 a $p < 0{,}01$)

Etiqueta	AUC_{CNN}	$AUC_{Trans.}$	z	p
background†	.648	.731	−4.12	3.8 e−5
christmas	.761	.712	+2.53	.011
commercial	.737	.703	+2.53	.011
dark†	.754	.714	+4.18	2.9 e−5
deep	.931	.910	+2.17	.030
documentary	.594	.632	−2.22	.027
dream†	.591	.554	+2.73	.006
emotional†	.660	.639	+3.12	.002
film†	.685	.718	−2.90	.004
game	.735	.677	+2.20	.028
groovy	.806	.744	+2.19	.029
happy	.657	.609	+2.54	.011
holiday†	.459	.705	−3.09	.002
melodic	.711	.658	+2.22	.026
movie†	.378	.230	+2.87	.004
positive†	.767	.651	+2.78	.005
retro	.593	.747	−2.01	.044
soundscape	.712	.763	−2.03	.042
uplifting	.585	.712	−2.22	.027

(c) Transformer vs. CRNN (12 / 55 etiquetas significativas; 5 a $p < 0{,}01$)

Etiqueta	AUC_{CRNN}	$AUC_{Trans.}$	z	p
background	.684	.731	−2.34	.020
ballad	.751	.787	−2.10	.036
commercial	.669	.703	−2.07	.039
dark†	.664	.714	−4.30	1.7 e−5
deep†	.944	.910	+3.63	.0003
documentary†	.575	.632	−3.48	.0005
happy	.641	.609	+2.17	.030
meditative	.768	.794	−2.06	.039
movie†	.359	.230	+2.80	.005
positive	.719	.651	+2.33	.020
relaxing†	.629	.583	+3.65	.0003
sport	.792	.850	−2.30	.022