eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Reconocimiento multimodal de emociones

Autora

Nahiara Romano Alonso

2023

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Computación

Trabajo de Fin de Grado

Reconocimiento multimodal de emociones

Autora

Nahiara Romano Alonso

Director

Juan Miguel López Gil

Resumen

En este Trabajo de Fin de Grado, se desarrollan y se prueban distintas implementaciones de redes neuronales para reconocer las emociones de las personas en tres modalidades diferentes: imagen, vídeo y audio. Se hacen pruebas con distintos tipos de redes neuronales y con distintas bases de datos públicas para saber qué red neuronal (y sus hiperparámetros), puede ser la más adecuada para cada modalidad. Una vez se obtienen los resultados del entrenamiento de estas redes, se comparan los resultados para saber cuál es la que ofrece mayor exactitud. Finalmente, desarrollar un algoritmo de fusión de clasificaciones de audio y vídeo.

En conclusión, podemos decir que los mejores resultados se han obtenido utilizando distintos tipos de redes neuronales convolucionales. En todas las bases de datos el mejor resultado se ha logrado utilizando diferentes arquitecturas CNNs, a excepción del reconocimiento del habla, que con una base de datos en concreto se ha alcanzado el mejor resultado utilizando una red neuronal recurrente.

Índice general

Resumen I

Índice general III

Índice de figuras VII

Índice de tablas XI

1. Introducción 1

1.1. Objetivos . . . 2

1.2. Herramientas utilizadas . . . 2

1.2.1. Anaconda . . . 2

1.2.2. Spyder . . . 2

1.2.3. Tensorflow . . . 3

1.2.4. Keras . . . 3

1.2.5. OpenCV . . . 3

1.2.6. Librosa . . . 3

1.2.7. Texmaker . . . 4

1.2.8. GitHub . . . 4

1.2.9. Hardware . . . 4

1.3. Estructura . . . 4

2. Planificación y gestión del proyecto **7**

2.1. Diagrama EDT 7

2.2. Desarrollo del proyecto 9

2.3. Periodos de realización de las tareas 9

2.3.1. Dedicación estimada 9

2.3.2. Análisis y gestión de riesgos 10

3. Estado del arte **11**

3.1. Reconocimiento facial 11

3.2. Reconocimiento en habla 13

4. Fundamentos teóricos **15**

4.1. Redes neuronales artificiales 15

4.2. Redes neuronales convolucionales 17

4.2.1. DCNN 19

4.2.2. Inception v3 21

4.2.3. AlexNet 22

4.3. Redes neuronales residuales 23

4.3.1. ResNet50 24

4.4. Redes neuronales recurrentes 25

4.4.1. LSTM 26

4.5. Clasificador MLP 27

4.6. Métricas de evaluación 27

4.6.1. Precision 27

4.6.2. Recall (exhaustividad 27

4.6.3. Accuracy (exactitud) 28

4.6.4. F1-Score 28

4.6.5. Matriz de confusión 28
4.7. Stratified k-fold cross validation 29
4.8. Fusión de modelos 30
4.8.1. Early fusion 30
4.8.2. Late fusion 31
4.8.3. Intermediate fusion 32

5. Desarrollo del proyecto **35**
5.1. Bases de datos utilizadas 35
5.1.1. FER-2013 36
5.1.2. RAVDESS 37
5.1.3. Crema-D 38
5.1.4. SAVEE 40
5.1.5. TESS 41
5.2. Procesamiento de los datos 42
5.2.1. Imagen 42
5.2.2. Vídeo 43
5.2.3. Audio 45
5.3. Extracción de características del audio 46
5.3.1. MFCC 47
5.3.2. Chroma 48
5.3.3. Mel Spectogram 48
5.3.4. STFT 49
5.4. Redes neuronales implementadas 50
5.4.1. AlexNet 50
5.4.2. VGG16 52

5.4.3. VGG19 54
5.4.4. Inception v3 56
5.4.5. CNN 56
5.4.6. CNN LSTM 57
5.4.7. LSTM 58
5.4.8. DCNN 59
5.4.9. ResNet50 61
5.5. Fusión 62
5.6. Resultados obtenidos 63
5.6.1. Imagen 63
5.6.2. Vídeo 64
5.6.3. Audio 66
5.6.4. Comparación de resultados 74

6. Desviaciones del proyecto **75**
6.1. Dedicación real 75
6.2. Desviación entre la dedicación estimada y la dedicación real 75

7. Conclusiones **77**

Bibliografía **79**

Índice de figuras

2.1. Diagrama EDT del Proyecto de Fin de Grado. 7

2.2. Diagrama de Gantt del Proyecto de Fin de Grado 9

4.1. Ejemplo de una red neuronal y sus diferentes capas: capa de entrada, capa oculta y capa de salida. 15

4.2. Ejemplo de una red neuronal y sus diferentes capas: capa de entrada, capa oculta y capa de salida. 16

4.3. Representación de las funciones de activación y sus fórmulas matemáricas. 17

4.4. Ejemplo de una CNN en la que se alternan capas de convolución y de pooling. 18

4.5. Ejemplos de Max Pooling y Average Pooling. 19

4.6. Arquitectura de VGG16. 20

4.7. Arquitectura de VGG19. 21

4.8. Arquitectura de Inception v3. 22

4.9. Arquitectura de AlexNet. 22

4.10. Error de entrenamiento (izquierda) y error de prueba (derecha) en CIFAR-10 con redes "simples"de 20 y 56 capas. La red más profunda tiene un mayor error de entrenamiento y, por tanto, de también de prueba. 23

4.11. Aprendizaje residual: un bloque de aprendizaje. 24

4.12. Arquitectura de una ResNet50. 25

4.13. Ejemplo de una red LSTM. 26

4.14. Fórmula matemática para calcular la precisión. 27
4.15. Fórmula matemática para calcular la exhaustividad. 27
4.16. Fórmula matemática para calcular la exactitud. 28
4.17. Fórmula matemática para calcular el F1-Score. 28
4.18. Ejemplo de una matriz de confusión. 29
4.19. Stratified 5 fold cross-validation 30
4.20. Early fusion. 31
4.21. Late fusion. 32
4.22. Intermediate fusion. 33

5.1. Distribución de las emociones en FER-2013. 36
5.2. Distribución de las emociones en RAVDESS. 38
5.3. Distribución de las emociones en Drema-D. 39
5.4. Distribución de las emociones en SAVEE. 41
5.5. Distribución de las emociones en TESS. 42
5.6. Algunas imágenes de FER-2013. 43
5.7. Algunos fotogramas extraídos de RAVDESS. 45
5.8. Representación del audio en forma de onda. 47
5.9. Comparación de una frase que representa enfado. 47
5.10. Comparación de una frase que representa alegría. 47
5.11. Ejemplo de las características MFCC de un archivo de audio. 48
5.12. Ejemplo de las características Chroma de un archivo de audio. 48
5.13. Ejemplo del Mel Spectogram de un archivo de audio. 49
5.14. Ejemplo del STFT de un archivo de audio. 49
5.15. Matriz de confusión del modelo AlexNet en FER-2013. 64
5.16. Matriz de confusión del modelo Inception v3 en RAVDESS. 65

5.17. Matriz de confusión del modelo CNN MFCC STFT en RAVDESS. 67

5.18. Matriz de confusión del modelo CNN MFCC Melspectogram en TESS. . 69

5.19. Matriz de confusión del modelo CNN MFCC en SAVEE. 71

5.20. Matriz de confusión del modelo AlexNet STFT en Crema-D. 73

Índice de tablas

2.1. Estimación en horas por cada bloque de tareas 9

5.1. Bases de datos utilizadas. 36

5.2. Resultados de los modelos de imagen en FER-2013 en tanto %. 63

5.3. Resultados de los modelos de vídeo en RAVDESS en tanto %. 64

5.4. Resultados en tanto % sobre base de datos RAVDESS. 66

5.5. Resultados de los modelos de audio en tanto % sobre base de datos TESS. 68

5.6. Resultados de los modelos de audio en tanto % sobre base de datos SAVEE. 70

5.7. Resultados de los modelos de audio en tanto % sobre base de datos Crema-D. 72

5.8. Comparación entre los mejores resultados registrados por otros autores y los resultados del proyecto. 74

6.1. Dedicación real en horas por cada bloque de tareas. 75

6.2. Desviación en horas por cada bloque de tareas. 76

1. CAPÍTULO

Introducción

Los avances que se han logrado durante los últimos en el campo de la Inteligenia Artificial, en gran medida se han llevado a cabo gracias al incremento de la capacidad computacional de la tecnología hardware. Estos avances permiten que en la actualidad sistemas de Inteligencia Artificial sean capaces de realizar tareas humanas con cierta complejidad de forma automática.

La Inteligencia Artificial se utiliza en sistemas reales en distintos problemas de variedad, así como la gestión y control, fabricación, ingeniería, educación, proceso de datos, finanzas, etc. También se utiliza en aplicaciones comerciales como en los diagnosticos médicos, monitorización durante la fabricación, análisis de problemas matemáticos, etc.

Una de las áreas que ha despertado más interés en la investigación en IA durante los últimos años es la affective computing[1], su objetivo es desarrollar sistemas capaces de interpretar, reconocer e incluso expresar sentimientos humanos.

Esta área se utiliza en muchas aplicaciones diferentes, por ejemplo, mejorando la experiencia del usuario en aplicaciones human-centric [3] y para la detección de polaridad en los comentarios en la red (distinción entre comentarios positivos o negativos) [4].

Por si esto fuera poco, esta disciplina también permite llevar a cabo un reconocimiento emocional a partir de las expresiones faciales. Esto se realiza a través de técnicas como Deep Learning (DL)[2] y el procesamiento del lenguaje natural (NLP)[3].

[1] https://www.media.mit.edu/groups/affective-computing/overview/
[2] https://www.ibm.com/topics/computer-vision?mhsrc=ibmsearch_a&mhq=computer%20vision%20
[3] https://www.ibm.com/topics/natural-language-processing?mhsrc=ibmsearch_a&mhq=nlp

En este proyecto se aplica la que en la actualidad del Deep Learning sea posiblemente la técnica de mayor importancia, las redes neuronales. Se aplican redes neuronales a los problemas de detección de reconocimiento de emociones en rostros humanos y en audios para obtener un modelo de representación robusto y fiable.

1.1. Objetivos

En este Trabajo de Fin de Grado, los objetivos planteados son los siguientes:

- Comprender e implementar distintos tipos de redes neuronales.
- Reconocer la emoción de una persona de una manera visual, a través de su cara.
- Reconocer la emoción de una persona de manera auditiva, a través de un archivo de audio.
- Contrastar resultados con distintas bases de datos públicas.
- Desarrollar un algoritmo de fusión para la clasificación de audio y vídeo.

1.2. Herramientas utilizadas

1.2.1. Anaconda

Anaconda es una distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático. Este software permite crear un entorno virtual para distintas versiones de Python y de de los paquetes que se quieran instalar.

1.2.2. Spyder

Spyder es un entorno de desarrollo integrado multiplataforma de código abierto para la programación científica en el lenguaje Python.

1.2.3. Tensorflow

Tensorflow es una biblioteca de software gratuita y de código abierto para el aprendizaje automático y la inteligencia artificial. Se puede utilizar en una gran variedad de tareas, pero tiene especial enfoque en el entrenamiento y la inferencia de redes neuronales profundas.

1.2.4. Keras

Durante el proyecto se ha utilizado la librería Keras sobre Tensorflow. Keras es una librería escrita en Python, diseñada específicamente para hacer experimentos con redes neuronales. Esta librería da la posibilidad de crear prototipos de una manera rápida ya que está pensada para que sea fácil de usar. Esta librería ha sido muy útil para el desarrollo del proyecto, se ha utilizado por ejemplo, para calcular la exactitud de los modelos implementados.

1.2.5. OpenCV

OpenCV es una biblioteca libre de visión artificial y aprendizaje automático, originalmente desarrollada por Intel. Esta librería se creó con el fin de proporcionar una infraestructura común para las aplicaciones de visión por ordenador y acelerar el uso de la percepción artificial en los productos comerciales.

La biblioteca cuenta con más de 2.500 algoritmos optimizados, que incluyen un amplio conjunto de algoritmos de visión por ordenador y aprendizaje automático. Estos algoritmos pueden utilizarse para detectar y reconocer caras, identificar objetos, clasificar acciones humanas en vídeos, rastrear movimientos de cámara, seguir objetos en movimiento, etc.

1.2.6. Librosa

Librosa es un paquete de Python que se utiliza para el análisis de audio y música. Este paquete se ha utilizado para reconocer las emociones en el audio y visualizar la representación del audio de manera gráfica.

1.2.7. Texmaker

Texmaker es un editor gratuito distribuido bajo la licencia GPL para escribir documentos de texto, multiplataforma, que integra muchas herramientas necesarias para desarrollar documentos con LaTeX.

1.2.8. GitHub

GitHub es una plataforma de alojamiento de código y desarrollo colaborativo que permite a los desarrolladores alojar y controlar versiones de su software. Es ampliamente utilizado por programadores individuales y equipos de todo el mundo para gestionar proyectos de código abierto y cerrado. Además de alojar y compartir código, GitHub también ofrece herramientas útiles para colaboración, seguimiento de problemas, documentación y mucho más. La implementación del proyecto se puede consultar en esta plataforma mediante el siguiente link https://github.com/nromano002/tfg.

1.2.9. Hardware

Para el desarrollo de este proyecto se ha utilizado un ordenador portátil HP Victus 16 –e0011ns con 16GB de memoria RAM, 512GB de disco duro y un procesador AMD Rayzen 7 5800H. Para entrenar las redes neuronales se ha utilizado la tarjeta gráfica NVIDIA GeForce RTX 3050.

1.3. Estructura

El proyecto está dividido en siete capítulos de la siguiente manera:

1. **Introducción del TFG**: Se presenta el contexto del proyecto, los objetivos principales del proyecto, las tecnologías utilizadas y la estructura de este documento.

2. **Planificación** y **gestión** del proyecto: Contiene los diagramas EDT y Gantt del proyecto, los distintos paquetes de trabajo del proyecto y el análisis de riesgos del proyecto y sus posibles soluciones.

3. **Estado del arte**: Visión general del estado del arte del reconocimiento de emociones en distintas modalidades. Se habla sobre otros trabajos previos y técnicas utilizadas.

4. **Fundamentos teóricos**: Se explican los conceptos necesarios para la correcta comprensión del proyecto y las distintas arquitecturas de las redes neuronales utilizadas.

5. **Desarrollo del proyecto**: Se exponen las bases de datos utilizadas, el procesamiento de los datos, la extracción de las características del audio, redes neuronales implementadas, métodos de fusión y resultados obtenidos.

6. **Desviaciones del proyecto**: Se presenta la dedicación real del proyecto en horas y la comparación ente la dedicación estimada y la dedicación real.

7. **Conclusiones**. Principales conclusiones del proyecto y futuras tareas que podrían realizarse.

2. CAPÍTULO

Planificación y gestión del proyecto

2.1. Diagrama EDT

El diagrama EDT correspondiente a este Trabajo de Fin de Grado se puede ver en la figura 2.1.

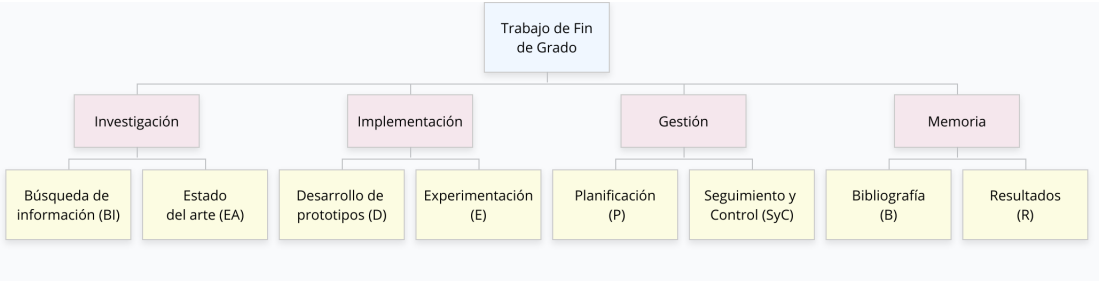


Figura 2.1: Diagrama EDT del Proyecto de Fin de Grado.

- El Paquete de Trabajo de Búsqueda de Información (BI) agrupará las tareas relacionadas con la búsqueda y estudio de información útil para el desarrollo del proyecto y para entender mejor el funcionamiento de los distintos tipos de algoritmos y redes neuronales existentes.
- El Paquete de Trabajo de Estado del arte (EA) agrupará las tareas del estudio del estado del arte del problema trabajado.
- El Paquete de Trabajo Desarrollo de prototipos (D) agrupará todas las tareas relacio-

nadas con el desarrollo y adaptación de los prototipos propios de redes neuronales para el reconocimiento de emociones en las distintas modalidades.

- El Paquete de Trabajo Experimentación (E) agrupará las tareas relacionadas con la experimentación de los prototipos existentes y con los prototipos creados con el principal objetivo de mejorar su rendimiento.
- El Paquete de Trabajo de Planificación (P) agrupará las tareas relacionadas con la planificación inicial y aquellas modificaciones que sean necesarias para mantener una planificación adecuada.
- El Paquete de Trabajo de Seguimiento y Control (SyC) agrupará las tareas relacionadas con el desarrollo del proyecto para asegurar su desarrollo y el cumplimiento de sus objetivos.
- El Paquete de Trabajo Bibliografía (B) agrupará las tareas necesarias para recopilar la información necesaria para la correcta comprensión de los fundamentos teóricos utilizados, los últimos avances y sus citaciones.s
- El Paquete de Trabajo Resultados (R) agrupará las tareas relacionadas con la organización de la memoria, su correcta redacción y los resultados obtenidos.

2.2. Desarrollo del proyecto

Los periodos de realización de los diferentes paquetes de trabajo se reflejan en la figura 2.2.

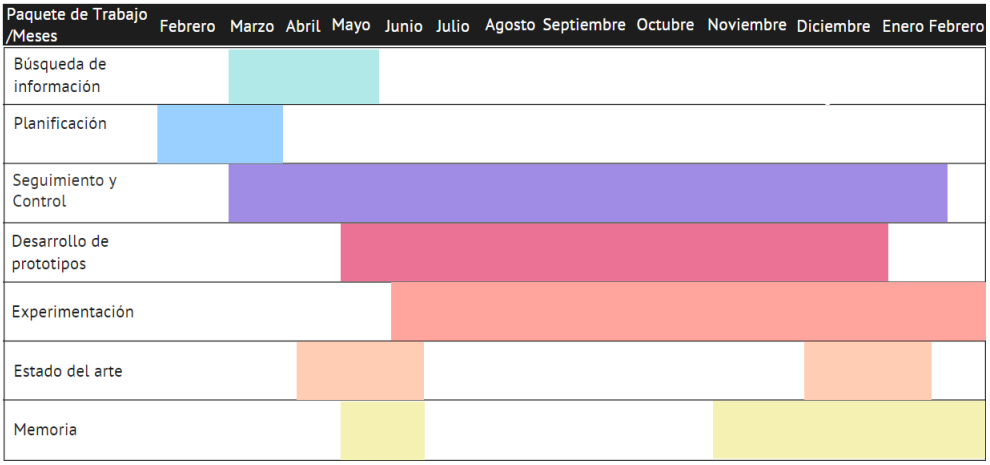


Figura 2.2: Diagrama de Gantt del Proyecto de Fin de Grado

2.3. Periodos de realización de las tareas

2.3.1. Dedicación estimada

En la siguiente tabla se muestra el tiempo estimado en horas para cada bloque de tareas:

Bloque	Bloque específico	Dedicación	Dedicación bloque
Investigación	Búsqueda de información Estado del arte	25h 45h	70h
Implementación	Desarrollo de prototipos Experimentación	75h 30h	105h
Gestión	Planificación Seguimiento y control	10h 30h	40h
Memoria	Bibliografía Resultados	45h 55h	100h
Total			315h

Tabla 2.1: Estimación en horas por cada bloque de tareas

2.3.2. Análisis y gestión de riesgos

Es posible que en el desarrollo del proyecto haya riesgos o inconvenientes que cambien totalmente el planteamiento del proyecto. Por este motivo, es muy importante reconocer los riesgos y tomar las medidas necesarias para afrontarlos.

A continuación se muestran los principales riesgos del proyecto y las medidas necesarias para evitarlos:

- **Riesgo 1**: Pérdida total o parcial de los archivos almacenados de manera local (datasets, modelos, código, ficheros de resultado...). Para evitar esto, en todo momento se almacenarán dos copias del trabajo realizado, una en la nube y otra en el equipo personal. De tal forma que en caso de perder una de ellas, se realizará otra copia en la mayor brevedad posible.

- **Riesgo 2**: Pérdida del entorno de Anaconda que se utiliza para desarrollar el proyecto. Este es un riesgo muy común, ya que en algunas ocasiones, algunos paquetes que se quieren instalar pueden ser incompatibles con alguna de las versiones de los paquetes ya instalados. En ese caso, se pueden actualizar o desactualizar los paquetes y provocar que el entorno virtual deje de funcionar. Para evitar este escenario, se clona el entorno utilizado. En caso de que uno de los entornos falle, se vuelve a clonar el entorno que no ha sido alterado a la mayor brevedad posible.

- **Riesgo 3**: Recursos y memoria limitada. En este proyecto se han utilizado distintos métodos de aprendizaje profundo con muchos datos y se han realizado múltiples pruebas con ellos, se ha invertido mucho tiempo en el entrenamiento de estos modelos. Por lo tanto, puede suceder que si los datos que se van procesar exceden la memoria disponible, no se pueda realizar el entrenamiento de las redes neuronales. Para evitar esto, cuando se entrenen las redes neuronales, se ejecutará el código mediante scripts secuenciales y se evitará tener muchas tareas en paralelo. En caso de que eso no sea suficiente, se utilizará Google Colab, es un producto de Google Research. Esta herramienta permite escribir y ejecutar código Python en el navegador y da la posibilidad de conectar conectarse a un entorno local o a un entorno virtual con más memoria RAM y GPUs más potentes.

3. CAPÍTULO

Estado del arte

El reconocimiento de emociones en habla y expresión facial ha sido un tema de interés en la psicología y la tecnología desde hace décadas. En la psicología, se han llevado a cabo numerosos estudios sobre la percepción de emociones en la voz y en la cara, y se ha desarrollado una teoría sobre cómo se codifican y decodifican las emociones en la comunicación humana. En la tecnología, el reconocimiento de emociones se ha utilizado en aplicaciones como la robótica, la inteligencia artificial y el análisis de sentimientos en las redes sociales. Con el avance de la tecnología, el reconocimiento de emociones se ha vuelto cada vez más preciso, lo que ha permitido su uso en una amplia gama de aplicaciones, incluyendo la medición de la satisfacción del cliente y la detección de trastornos emocionales.

3.1. Reconocimiento facial

El reconocimiento facial es una tarea complicada incluso para los seres humanos ya que solo es posible recordar y reconocer un número limitado de caras. Una de las herramientas más importantes que tenemos los humanos para expresar emociones son las expresiones faciales. Es por ello que el problema de reconocimiento automático de las expresiones faciales lleva investigándose desde hace más de 25 años.

A lo largo de estos años, se han ido desarrollando y probando distintas técnicas de aprendizaje automático para reconocer las expresiones faciales. Para la clasificación se utilizan algoritmos clásicos de aprendizaje automático, como por ejemplo Support Vector Ma-

chine (SVM) o LDA. En cuanto a la extracción de características, una de las primeras aproximaciones se basa en utilizar Gabor wavelets [12]. Estas son funciones que simulan la percepción del sistema visual humano. Otra técnica utilizada es Local Binary Patterns (LBP), como se puede ver en [13], que además se muestra una comparación demostrando una mejoría en la precisión del sistema.

Sin embargo, estos modelos tienen algunos problemas. El primero es que puede requerir la ayuda de un humano ya que es un proceso semi-automático. El segundo es que la precisión del sistema depende de los dos modelos, esto dificulta su optimización.

Estos modelos quedaron obsoletos cuando llegaron las redes neuronales artificiales. Una red neuronal es un sistema totalmente automático, rápido, que no necesita la extracción de características y mejora considerablemente los resultados. Debido a estos motivos, en la actualidad, la mayoría de los sistemas de reconocimiento de expresiones faciales utilizan estas redes.

FER-2013 fue diseñado por Goodfellow et al. como una competición de Kaggle para promover el desarrollo de mejores sistemas en el reconocimiento del habla. Los tres mejores equipos utilizaron CNN entrenadas con transformaciones de imagen [15]. El ganador, Yichuan Tang, alcanzó una exactitud del 71,2 % utilizando el objetivo primal de una SVM como función de pérdida para el entrenamiento y, además, utilizó la función de pérdida L2-SVM [16]. En su momento fue una novedad y dio muy buenos resultados en el conjunto de datos del concurso (FER-2013). FER-2013 es una base de datos pero también un concurso de Kaggle ya que en el concurso se reconocen las emociones de las personas en imágenes de la base de datos FER-2013 y por ello, recibió el mismo nombre.

Existen numerosas investigaciones en el ámbito del reconocimiento facial de emociones. En concreto, hay un estudio reciente de W. Deng sobre el estado actual del reconocimiento de emociones en habla basado en el aprendizaje profundo [17]. En otro artículo de Pramerdorfer y Kampel [18] se describen los enfoques adoptados por seis artículos actuales y recopila sus redes para lograr una exactitud del 75,2 % en la base de datos FER-2013. Para lograr ese resultado se utilizó una CNN.

Entre los seis artículos, Zhang et al. lograron la mayor una exactitud del 75,1 % (la mayor). Para ello, utilizaron datos auxiliares y características adicionales: se calculó un vector de características HoG[1] de parches faciales y se procesaron por la primera capa Fully Connected de la CNN (fusión temprana). También emplearon el registro facial, que demuestra sus ventajas incluso en condiciones difíciles (la extracción de puntos de referencia facia-

[1] https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f

les es imprecisa en un 15 % de las imágenes del conjunto de datos FER-2013) [19]. El siguiente artículo con mejores resultados es Kim et al, utilizó el registro facial, el aumento de registro facial, aumento de datos, características adicionales y ensamblaje [20].

Hasta el momento, el mayor resultado registrado es del 76.82 % de exactitud en la base de datos FER-2013. Para ello, se ha utiliza ensamblaje de una ResMaskingNet con otras 6 CNNs [21].

En cuanto a la base de datos RAVDESS, el mejor resultado se alcanzó en 2022 (86,70 % de exactitud) utilizando un transformador de fusión intermedio [22].

3.2. Reconocimiento en habla

Cuando hablamos de reconocimiento de las emociones en voz, no nos referimos al qué se dice sino al cómo se dice. En general, para el análisis de emociones en datos acústicos, primero es necesario extraer las características generales que están relacionadas con el tono, la energía y la duración. También se utilizan otras características más complejas, como los Mel Frequency Cepstral Coefficients (MFCC).

Los últimos estudios utilizan la potencia de las redes neuronales para la clasificación del audio. Por ejemplo, en [14] se hace un exhaustivo estudio en el que se compara el desempeño de los diferentes tipos de redes neuronales en la tarea de reconocimiento de emociones a partir de audio.

Las características extraídas son bidimensionales, por lo tanto, es posible utilizar redes densas normales aplanando las características, redes convolucionales, o redes recurrentes. Aunque a simple vista la LSTM puede parecer la opción más lógica, ya que procesa secuencias con una relación temporal, la red neuronal convolucional ofrece resultados realmente buenos.

Las DNNs [23] y DCNNs [24], redes más profundas que las CNN, fueron los dos primeros enfoques de aprendizaje profundo Por lo tanto, es un paso justo examinar los enfoques del reconocimiento de emociones del habla del aprendizaje profundo. Por ejemplo, en la clasificación de emociones la red neuronal [25] se utilizó para aprender características de alto nivel a partir de los rasgos de bajo nivel derivados a nivel acústico. acústico. Las DNN fueron algunas de las primeras en probarse. Pero el estudio anterior no tuvo debidamente en cuenta la eliminación adecuada de las características, por lo que los resultados fueron poco fiables.

En el reconocimiento de emociones en el habla, los mejores resultados se han logrado con modelos de deep learning que combinan características tanto acústicas como prosódicas de la voz. Por ejemplo, se ha demostrado que las redes neuronales que utilizan características de la señal de voz como la energía, la frecuencia fundamental y los formantes, junto con la información prosódica, como la entonación y la duración de las palabras, pueden lograr una precisión de clasificación de emociones cercana al 90 %.

Además, la utilización de técnicas de aprendizaje profundo como las redes neuronales de atención y las redes LSTM ha mejorado aún más los resultados, al permitir que los modelos puedan considerar la secuencia temporal de las características de la voz para una mejor detección de las emociones.

Sin embargo, es importante destacar que el reconocimiento de emociones en el habla es un problema complejo y aún hay desafíos por superar, como la variabilidad interpersonal en la expresión emocional en la voz, y la necesidad de una mayor cantidad de datos etiquetados para entrenar los modelos.

A. U y K V K. [26] han utilizado redes neuronales convolucionales y han entrenado el modelo combinando bases de datos (por ejemplo, RAVDESS y TESS). Mediante aumento de datos han podido mejorar los resultados obtenidos, hasta un 68 % de exactitud combinando ambas bases de datos. Haciendo una distinción de género, lograron aumentar ese porcentaje hasta el 75 % en RAVDESS. Finalmente, proponen un modelo que logra el 89 % de exactitud combinando las bases de datos TESS y RAVDESS y distintas técnicas de aumento de datos.

En la base de datos RAVDESS el mejor resultado alcanzado fue en 2022, alcanzando un 92,08 % de exactitud, utilizando Temporal-aware bI-direction Multi-scale Network (TIM-Net) [27].

En la base de datos Crema-D, se ha logrado un 70,95 % de exactitud combinando los modelos SepTr y Learning Rate Curriculum (LeRaC) [28].

En cuanto a la base de datos TESS el mayor resultado registrado es de un 97,2 % de exactitud. Para ello, en el artículo [29] se menciona la utilización de los clasificadores Support Vector Machine (SVM) y MultiLayer Perceptron (MLP).

Finalmente, el mejor resultado en la base de datos SAVEE se logró en 2022 utilizando una TIM-Net [27], alcanzando una exactitud del 87,71 %.

4. CAPÍTULO

Fundamentos teóricos

4.1. Redes neuronales artificiales

Una red neuronal es un modelo computacional que trata de simular el funcionamiento del cerebro humano. Consiste en un conjunto de nodos, llamados neuronas artificiales, conectados entre sí para transmitir señales. Las señales son emitidas desde la entrada hasta generar una salida. El principal objetivo de este modelo es aprender modificándose automáticamente a si mismo, de forma que pueda realizar tareas más complejas que las que se podrían hacer mediante la programación básica basada en reglas. Gracias a las redes neuronales se pueden automatizar funciones que no podrían ser llevadas a cabo por personas.

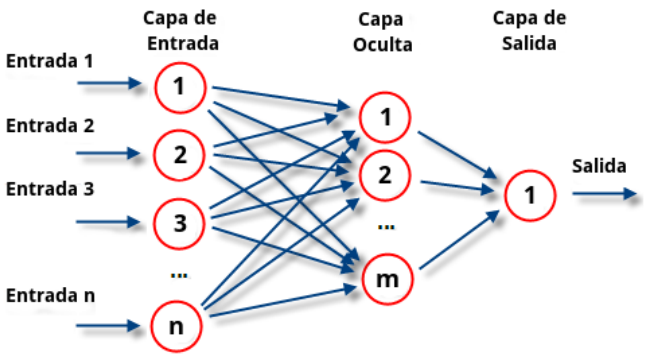


Figura 4.1: Ejemplo de una red neuronal y sus diferentes capas: capa de entrada, capa oculta y capa de salida.

Cada neurona está conectada con otras a través de unos enlaces que contienen un peso. Como se puede observar en la figura 4.2, la neurona tiene cinco entradas (x1,x2,x3,x4,x5). Cada una de las entradas se conectan a través de cinco distintos enlaces y cada enlace tiene asociado un peso (w1,w2,w3,w4,w5). Multiplicando cada valor de entrada con su correspondiente peso se obtiene la entrada de la neurona que se está tratando. Normalmente, la salida es equivalente al valor resultante de la función de activación.

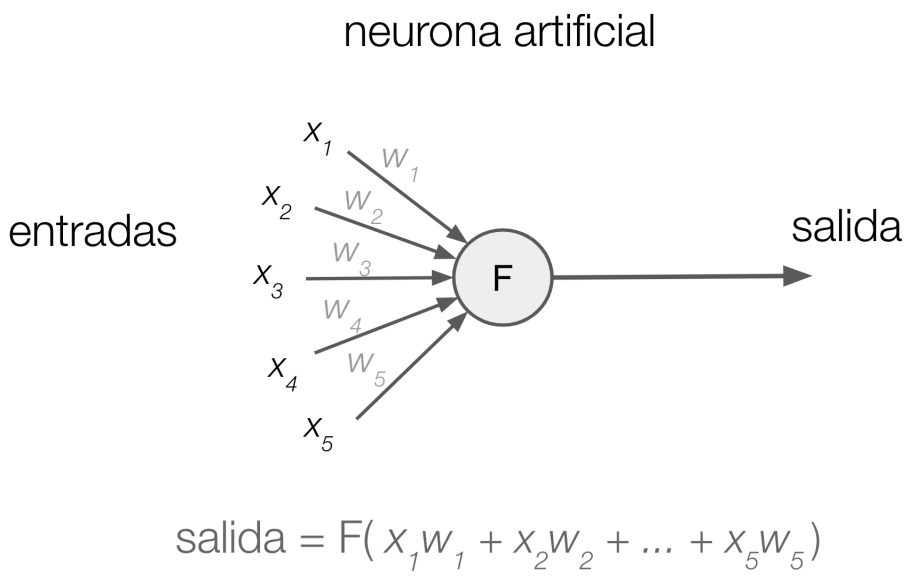


Figura 4.2: Ejemplo de una red neuronal y sus diferentes capas: capa de entrada, capa oculta y capa de salida.

Cada neurona recibe como entrada la salida de las neuronas de la capa anterior multiplicada por el peso de su correspondiente enlace. Después se aplica la función de activación a la información obtenida para generar la salida.

La función de activación es una función que transmite la información creada por la combinación lineal de los pesos y las entradas; son la forma de transmitir la información a través de las conexiones de salida.

Existen distintas funciones de activación pero las más utilizadas son las siguientes:

- Función **ReLU**: Transforma los valores introducidos anulando los valores negativos y dejando los positivos tal y como entran.
- Función **sigmoid**: La función sigmoide transforma los valores introducidos a una escala (0,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0.

- Función **tangente hiperbólica**: Transforma los valores introducidos a una escala (-1,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a -1.

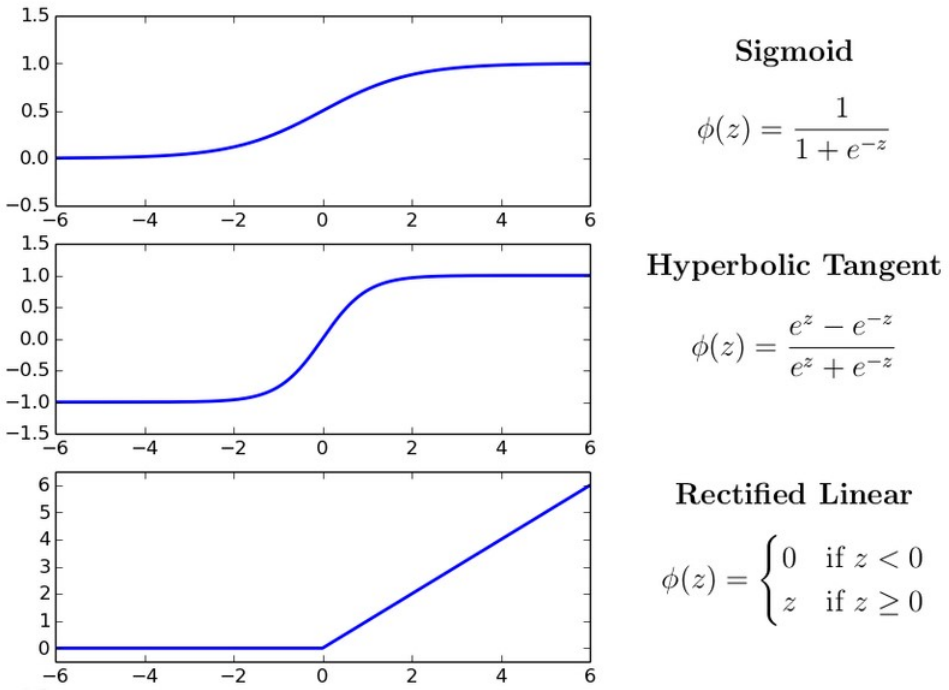


Figura 4.3: Representación de las funciones de activación y sus fórmulas matemáricas.

4.2. Redes neuronales convolucionales

La Convolutional Neural Network (CNN) es un tipo de red neuronal artifical donde los nodos corresponden a campos receptivos de una manera muy similar a las neruronas en la corteza visual primaria de un cerebro biológico, y que son variaciones de un perceptrón multicapa, que implementan convoluciones sobre la capa de entrada, de ahí el nombre genérico de estas redes.

La aplicación de estas redes es realizada en matrices bidimensionales, por lo que son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras.

Las redes convolucionales se basan en una combinación de capas de convolución y de pooling. Normalmente, al final de la red suele haber una capa totalmente conectada (fully connected layer). Esta última capa se encarga de realizar la clasificación.

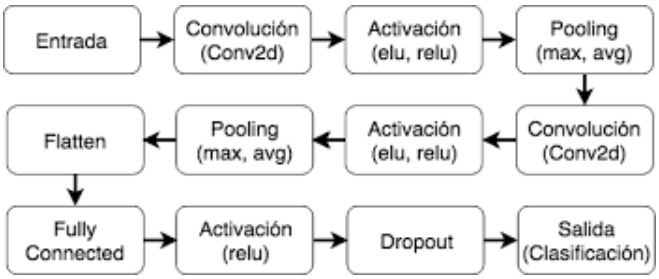


Figura 4.4: Ejemplo de una CNN en la que se alternan capas de convolución y de pooling.

A continuación se explican las distintas capas de una CNN:

- Capa de **convolución**: En una capa de convolución se aplican diferentes filtros sobre una imagen. La profundidad de la capa convolucional define en número de filtros que se aplicarán sobre cada imagen, cada filtro se utilizará para identificar un patrón distinto. Después de pasar por esta capa, el tamaño de la imagen puede disminuir.

- Capa de **Pooling**: El objetivo de esta capa es disminuir el tamaño de la imagen para reducir la cantidad de enlaces en la red y evitar el overfitting. Principalmente, se diferencian dos tipos: el average pooling y el max pooling. Ambos tipos tienen el mismo comportamiento (una ventana de la imagen se sustituye por un único valor calculado sobre ella). Sin embargo, su funcionamiento es diferente: el average pooling calcula la media mientras que el max pooling indica el máximo (como indican sus nombres).

- **Dropout**: Es una técnica de regularización simple y poderosa para redes neuronales y modelos de aprendizaje profundo. Un buen valor para el Dropout en una capa oculta está entre 0,5 y 0,8. Las capas de entrada utilizan una tasa de Dropout mayor, como 0,8.

- **Dense**: Es la única capa de red real en el modelo. Alimenta todas las salidas de la capa anterior a todas sus neuronas, cada neurona proporciona una salida a la siguiente capa. Dense(1024) significa que tiene 1024 neuronas.

- **Batch Normalization**: Normaliza los datos de la capa anterior. Batch Normalization es una técnica de normalización utilizada en entrenamiento de redes neuronales artificiales. Su objetivo es normalizar la activación de cada capa en una red neuronal, lo que ayuda a prevenir el problema de la desviación de la distribución de los pesos durante el entrenamiento. Esto a su vez mejora la velocidad de convergencia y reduce la varianza en los resultados del modelo, lo que a menudo resulta en un

mejor rendimiento en la tarea de aprendizaje automático. Batch Normalization es aplicado a la salida de cada capa antes de pasar a la siguiente capa y es esencialmente una técnica de regularización.

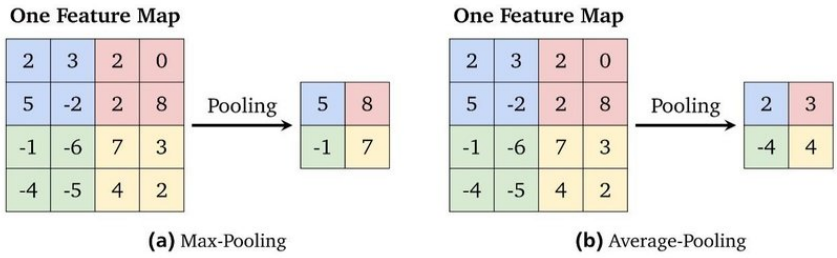


Figura 4.5: Ejemplos de Max Pooling y Average Pooling.

4.2.1. DCNN

La ventaja de las Deep Convolutional Neural Network (DCNN) reside en su estratificación. Una DCNN utiliza una red neuronal tridimensional para procesar los elementos rojo, verde y azul de la imagen al mismo tiempo. Esto reduce considerablemente el número de neuronas artificiales necesarias para procesar una imagen, en comparación con las redes neuronales tradicionales. Una VGG es una red neuronal convolucional profunda propuesta por K. Simonyan y A. Zisserman, de la Universidad de Oxford. Esta red neuronal consiguió especial reconocimiento al ganar el Desafío de Reconocimiento Visual a Gran Escala de ImageNet[1] (ILSVRC) en 2014. El modelo alcanzó una precisión del 92,7 % en la base de datos Imagenet, una de las puntuaciones más altas logradas.

Entre todas las configuraciones que se han probado a lo largo de los años, se ha demostrado que VGG16 es el modelo con mejor rendimiento en la base de datos de ImageNet.

Una VGG16 es una red neuronal convolucional con 16 capas de profundidad. Esta red neuronal puede cargar una versión pre-entrenada de la red entrenada en más de un millón de imágenes desde la base de datos de ImageNet.

[1] https://www.image-net.org/

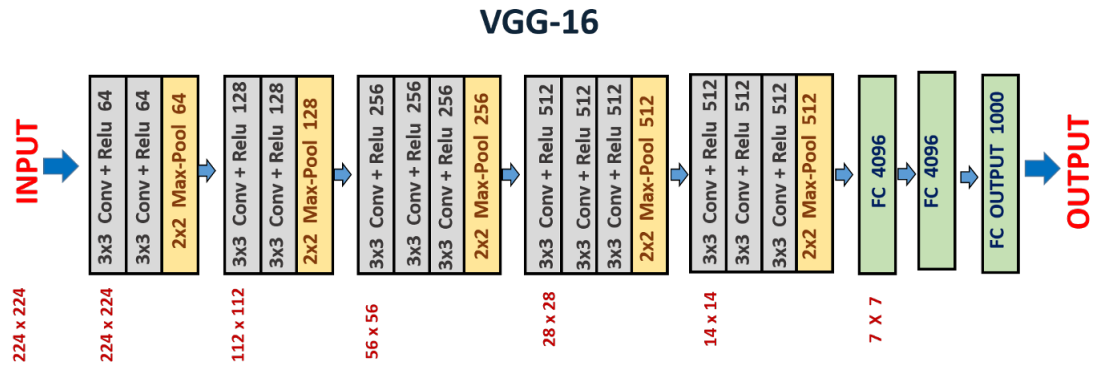


Figura 4.6: Arquitectura de VGG16.

- VGG16 toma el tamaño del tensor de entrada como 224x 244 con 3 canales RGB (224,224,3).
- Lo más destacable de VGG16 es que en lugar de tener un gran número de hiperparámetros se centraron en tener capas de convolución de filtro 3x3 con stride[2] 1 y siempre usaron la misma capa de Padding y Maxpool de filtro 2x2 de stride 2. La primera capa de convolución tiene 64 filtros, la segunda tiene 128 filtros, la tercera tiene 256 filtros y las últimas dos capas convolucionales tienen 512 filtros.
- Tres capas totalmente conectadas (FC) siguen a una pila de capas convolucionales: las dos primeras tienen 4096 canales cada una, la tercera realiza una clasificación ILSVRC[3] de 1000 vías y, por tanto, contiene 1000 canales (uno para cada clase). La última capa es la capa soft-max.

De la misma manera, VGG19 es una red neuronal convolucional con 19 capas de profundidad. Tal y como sucede con VGG16, esta red neuronal puede cargar una versión pre-entrenada de la red entrenada en más de un millón de imágenes desde la base de datos de ImageNet.

Como se puede observar en la figura 4.7, la VGG19 tiene 16 capas de convolución agrupadas en 5 bloques. Después de cada bloque, hay una capa Maxpool que disminuye el tamaño de la imagen de entrada en 2 y aumenta el número de filtros de la capa de convolución también en 2. Las dimensiones de las tres últimas capas densas del bloque 6 son 4096, 4096 y 1000 respectivamente.

[2]https://deepai.org/machine-learning-glossary-and-terms/stride
[3]https://www.image-net.org/challenges/LSVRC/

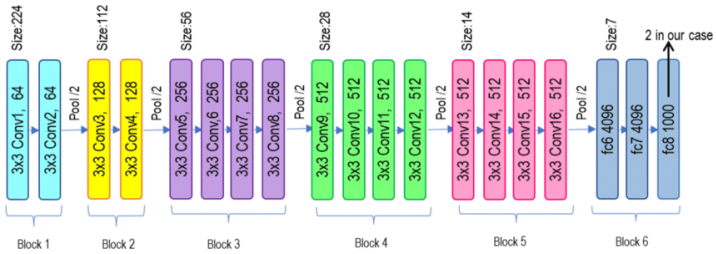


Figura 4.7: Arquitectura de VGG19.

4.2.2. Inception v3

El modelo Inception v3 fue lanzado en el año 2015, tiene un total de 42 capas y una tasa de error más baja que sus predecesores. En Inception v3 se hace uso de unos módulos llamados Inception. Estos módulos actúan como múltiples filtros aplicados a un mismo valor de entrada mediante capas convolucionales y de pooling. Después, el resultado de estos filtros es concatenado y utilizado como el valor de salida del módulo. Este modelo aumenta el número de parámetros entrenables y la computación requerida, pero mejora considerablemente la precisión.

Las principales modificaciones en el modelo Inception v3 respecto a sus versiones anteriores:

- Factorización en convoluciones más pequeñas.
- Factorización espacial en convoluciones asimétricas.
- Utilidad de los clasificadores auxiliares.
- Reducción eficiente del tamaño de la cuadrícula.

Inception v3 es un modelo pre-entrenado de 48 capas de profundidad. Es una versión de la red ya entrenada con más de un millón de imágenes de la base de datos ImageNet. La red tiene un tamaño de entrada de imagen de 299 por 299. El modelo extrae características generales de las imágenes de entrada en la primera parte y las clasifica basándose en ellas en la segunda.

Este modelo es ampliamente utilizado en el reconocimiento de imágenes, se ha demostrado alcanzar una precisión superior al 78,1 % en el conjunto de datos ImageNet y en torno al 93,9 % de precisión en los 5 mejores resultados.

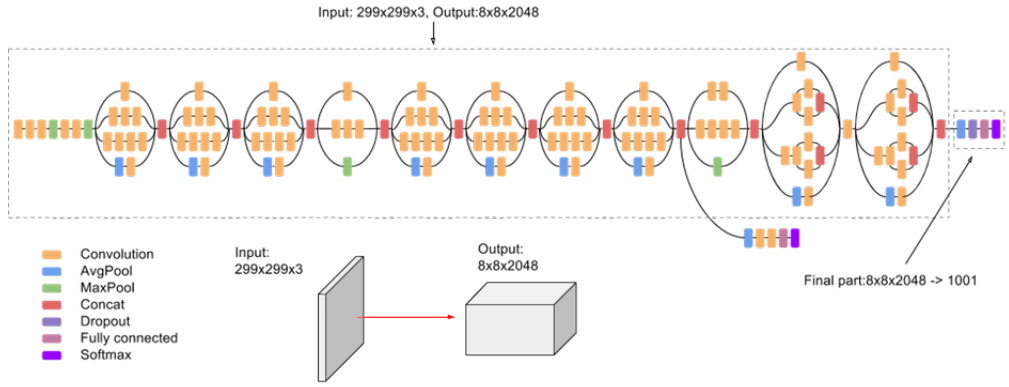


Figura 4.8: Arquitectura de Inception v3.

4.2.3. AlexNet

AlexNet es una red neuronal convolucional con 8 capas de profundidad. Como las redes mencionadas anteriormente, puede cargar una versión pre-entrenada de la red entrenada sobre la base de datos ImageNet.

Como se puede observar en la figura 4.9 la arquitectura de esta red está formada por 5 capas convolucionales, de las cuales la primera, la segunda y la quinta tienen capas Max-Pooling para la extracción adecuada de las características. Las capas Max-Pooling están superpuestas y tienen srides de 2 con un tamaño de filtro de 3×3. El resultado es una reducción de los porcentajes de error en los primeros 1 y 5 puestos del 0,4 % y el 0,3 %, respectivamente. De este modo, las tasas de error de las capas 1 y 5 superiores disminuyen un 0,4 % y un 0,3 %, respectivamente, en comparación con las capas Max-Pooling no superpuestas. A continuación, hay dos capas totalmente conectadas (cada una con abandono) y una capa softmax al final para las predicciones.

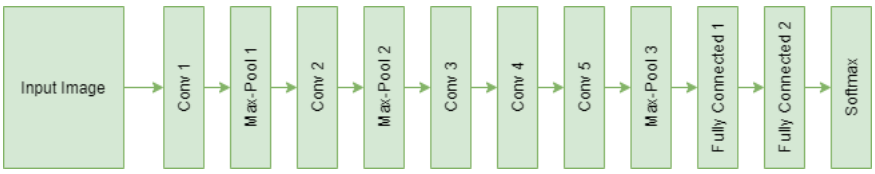


Figura 4.9: Arquitectura de AlexNet.

4.3. Redes neuronales residuales

En 2012, Krizhevsky et al. [1] pusieron el foco en las redes neuronales convolucionales profundas [2]. Fue la primera vez que esta arquitectura tuvo más éxito que el aprendizaje de características tradicional y manual en ImageNet. Su DCNN, llamada AlexNet, contenía 8 capas de red neuronal, 5 convolucionales y 3 totalmente conectadas. Esto estableció los fundamentos de la CNN tradicional, una capa convolucional seguida de una función de activación seguida de una operación de agrupación máxima.

Gran parte del éxito de las redes neuronales profundas se debe a estas capas adicionales. La intuición que subyace a su función es que estas capas aprenden progresivamente características más complejas. La primera capa aprende los bordes, la segunda las formas, la tercera los objetos, la cuarta los ojos, y así sucesivamente.

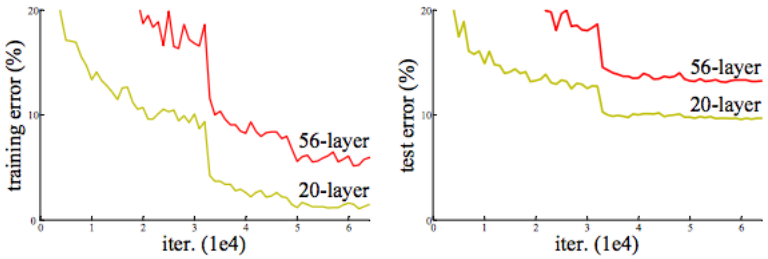


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Figura 4.10: Error de entrenamiento (izquierda) y error de prueba (derecha) en CIFAR-10 con redes "simples"de 20 y 56 capas. La red más profunda tiene un mayor error de entrenamiento y, por tanto, de también de prueba.

Como se puede observar en la figura 4.10, He et al. [1] representan el error de entrenamiento y training de una CNN de 20 capas frente a una CNN de 56 capas. Este gráfico desafía nuestra creencia de que añadir más capas crearía una función más compleja, por lo que el fallo se atribuiría al sobreajuste. Si éste fuera el caso, los parámetros de regularización adicionales y los algoritmos como dropout o L2-norms serían un enfoque exitoso para arreglar estas redes. Sin embargo, el gráfico muestra que el error de entrenamiento de la red de 56 capas es mayor que el de la red de 20 capas, lo que pone de manifiesto un fenómeno diferente que explica su fallo.

El fracaso de la CNN de 56 capas podría deberse a la función de optimización, a la inicialización de la red o al famoso problema del gradiente desvanecido. Sin embargo, los autores argumentan que el uso de la normalización garantiza que los gradientes tengan normas correctas. Entre las muchas teorías que explican por qué las redes más profundas no funcionan mejor que sus homólogas superficiales, a veces es mejor buscar una explicación en los resultados empíricos y trabajar hacia atrás a partir de ahí. El problema de entrenar redes muy profundas se ha aliviado con la introducción de una nueva capa de red neuronal: el bloque residual.

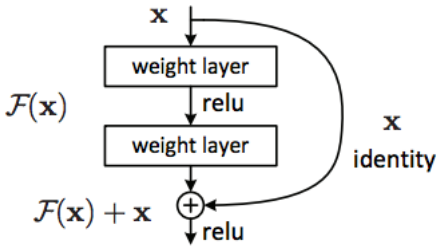


Figura 4.11: Aprendizaje residual: un bloque de aprendizaje.

La modificación más importante a comprender es la conexión de salto, mapeo de identidad. Este mapeo de identidad no tiene ningún parámetro y sólo está ahí para añadir la salida de la capa anterior a la capa de adelante. Sin embargo, a veces x y F(x) no tendrán la misma dimensión. Recordemos que una operación de convolución suele reducir la resolución espacial de una imagen, por ejemplo, una convolución 3x3 en una imagen de 32 x 32 da como resultado una imagen de 30 x 30. El mapeo de identidad se multiplica por una proyección lineal W para expandir los canales de atajo para que coincidan con el residual. Esto permite combinar la entrada x y F(x) como entrada a la siguiente capa.

4.3.1. ResNet50

ResNet50 es una red neuronal re profunda con 50 capas de profundidad. Esta red basada en bloques residuales fue presentada por Microsoft en 2015. Esta red está disponible en distintos tamaños a través de Keras Applications[4], la mayor de ellas es de 152 capas, que consiguió tener menos complejidad que VGGNet[5].

[4]https://keras.io/api/applications/
[5]https://deepchecks.com/glossary/vggnet/

El modelo ResNet50 consta de 5 etapas, cada una con un bloque de convolución y otro de identidad. Cada bloque de convolución tiene 3 capas de convolución y cada bloque de identidad también tiene 3 capas de convolución. ResNet50 tiene más de 23 millones de parámetros entrenables.

La idea de ResNet reside en preguntarse por qué apilar capas y capas, se sabe que la red no mejora a partir de cierta profundidad. En este sentido aparecen problemas como el desvanecimiento de gradiente (vanishing gradient)[6]. De la misma manera, con la profundidad de las capas llega un momento en el que la red se estanca y comienza a degradarse (overfitting)[7].

Por lo tanto, se deduce que las redes convolucionales menos profundas en ocasiones aprenden mejor que las redes más profundas. Es por ello que surge la idea de los bloques residuales, estos permiten saltar unas capas donde se aumenta el número de capas introduciendo una conexión residual (con una capa identidad), mejorando el proceso de aprendizaje.

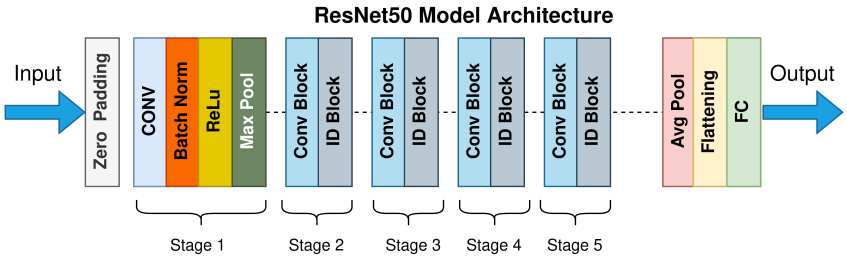


Figura 4.12: Arquitectura de una ResNet50.

4.4. Redes neuronales recurrentes

Las Recurrent Neural Network (RNNs) son un tipo de red neuronal capaz de procesar secuencias de de longitud variable. Por lo tanto, son muy útiles en el procesamiento del audio ya que sirven para analizar datos de series temporales permitiendo tratar la dimensión del tiempo. Una red neuronal recurrente transmite datos hacia adelante pero también hacia atrás. De tal forma que, a medida que la red avanza, obtiene información de las anteriores neuronas y de ella misma en el paso previo.

[6] https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484
[7] https://www.ibm.com/topics/overfitting?mhsrc=ibmsearch_a&mhq=overfitting

Estas redes neuronales permiten que la aplicación de la que forman parte pueda recordar y olvidar lo que va analizado. Por esta razón, se dice que son redes que tienen memoria y pueden analizar datos recibidos en tiempos distantes. Por lo tanto, es un modelo que va almacenando información que procesó al inicio de la secuenciación y la vincula con los nuevos datos que va recibiendo.

4.4.1. LSTM

Una red Long Short Term Memory (LSTM) es un tipo de red neuronal recurente capaz de recordar un dato relevante de la secuencia y retenerlo por varios instante. Por lo tanto, puede tener una memoria a corto plazo (como las redes recurrentes básicas) y a largo plazo.

Funciona de manera similar a como lo haría un cerebro humano para analizar las secuencias. Si por ejemplo, queremos comprar un ordenador, leemos algunas valoraciones escritas por distintos compradores. Sin embargo, a la hora de tomar la decisión no nos enfocamos en todo el texto sino en las palabras que consideramos más relevantes y descartamos la demás información.

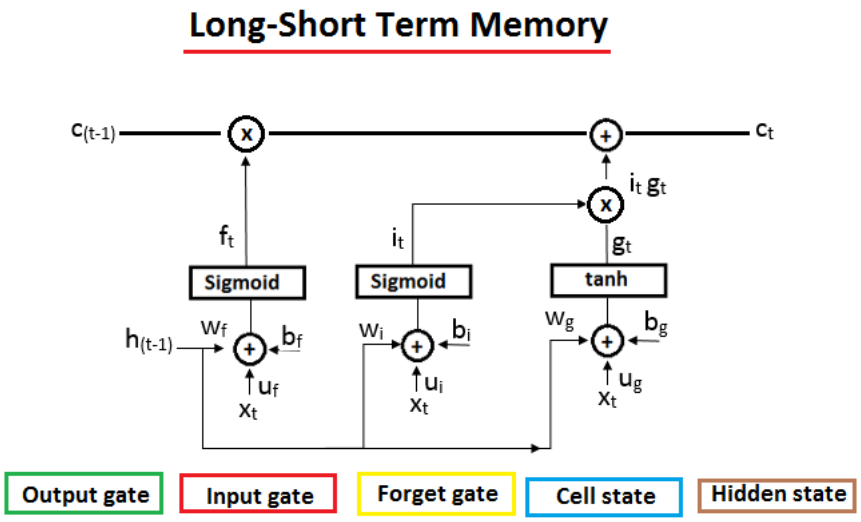


Figura 4.13: Ejemplo de una red LSTM.

4.5. Clasificador MLP

Multilayer Perceptron (MLP) es un tipo de modelo de aprendizaje automático de redes neuronales artificiales. Es un tipo de red neuronal feedforward que consta de varias capas de nodos, conectados por pesos y utilizados para clasificar o realizar regresión en conjuntos de datos. Se aplica en una amplia variedad de tareas, incluyendo reconocimiento de imágenes, procesamiento de lenguaje natural y detección de fraudes.

4.6. Métricas de evaluación

Algunas de las métricas que se utilizan para evaluar el rendimiento en clasificación son las siguientes: precision, recall, accuracy, F1-score, y la matriz de confusión.

4.6.1. Precision

Es la proporción de resultados predecidos correctamente con respecto a todas las predicciones. También se conoce como sensibilidad o especificidad.

$$precision = \frac{TP}{TP + FP}$$

Figura 4.14: Fórmula matemática para calcular la precisión.

4.6.2. Recall (exhaustividad

Nos informa sobre la cantidad que el modelo de machine learning es capaz de identificar.

$$recall = \frac{TP}{TP + FN}$$

Figura 4.15: Fórmula matemática para calcular la exhaustividad.

4.6.3. Accuracy (exactitud)

Es la proporción de predicciones correctas entre todas las predicciones realizadas por un algoritmo.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Figura 4.16: Fórmula matemática para calcular la exactitud.

4.6.4. F1-Score

Combina las anteriores tres métricas en una sola que oscila entre 0 y 1 y tiene en cuenta tanto la Precisión como el Recall.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figura 4.17: Fórmula matemática para calcular el F1-Score.

4.6.5. Matriz de confusión

Nos permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

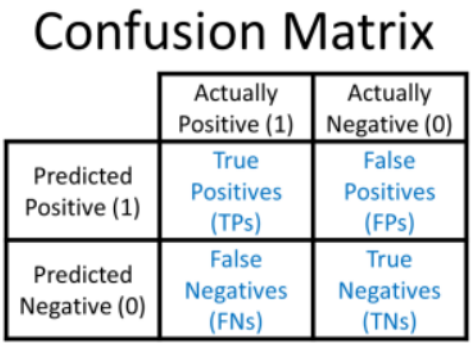


Figura 4.18: Ejemplo de una matriz de confusión.

4.7. Stratified k-fold cross validation

La validación cruzada es un método que consiste en evaluar el rendimiento de un modelo de machine learning, con el fin de encontrar un mejor modelo rápidamente. Stratified k-fold cross validation es una extensión de k-fold cross-validation. La diferencia es que, en lugar de que las divisiones de los conjuntos sea completamente aleatoria, se mantiene la proporción de cada clase en cada subconjunto. Los datos se distribuyen en k subconjuntos (folds). Normalmente, se usa con k=5 o k=10 folds, para el desarrollo del proyecto de ha utilizado k=5.

En esta técnica, el modelo se entrena utilizando k-1 subconjuntos en el conjunto de entrenamiento. El modelo resultante se valida se valida con la parte restante de los datos (es decir, se utiliza como conjunto de prueba para calcular una medida de rendimiento como la exactitud).

La medida de rendimiento por lo tanto, es la media de los valores calculados en el bucle. Este enfoque puede ser costoso desde el punto de vista computacional, pero no desperdicia demasiados datos (como ocurre cuando se fija un conjunto de validación arbitrario), lo que supone una gran ventaja en problemas como la inferencia inversa, en los que el número de muestras es muy reducido.

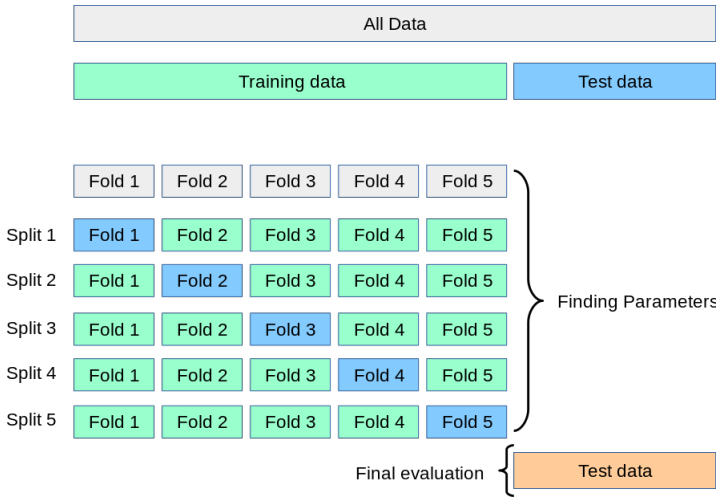


Figura 4.19: Stratified 5 fold cross-validation

4.8. Fusión de modelos

A continuación se explicarán las tres técnicas de fusión de datos multimodales [8] [9] que se utilizan en la actualidad.

4.8.1. Early fusion

La fusión temprana (también conocida como fusión a nivel de datos) es una forma tradicional de fusionar múltiples datos antes de realizar el análisis, como se puede observar en la figura 4.20.

Este método se denomina fusión a nivel de entrada. La investigación [9] propone dos posibles enfoques para la técnica de fusión temprana. El primer enfoque consiste en combinar los datos eliminando la correlación entre dos sensores. El segundo enfoque consiste en fusionar los datos en su espacio común de dimensión inferior. Existen muchas soluciones estadísticas que pueden utilizarse para llevar a cabo uno o ambos métodos, como el análisis de componentes principales (PCA), el análisis de correlación canónica y el análisis de componentes independientes.

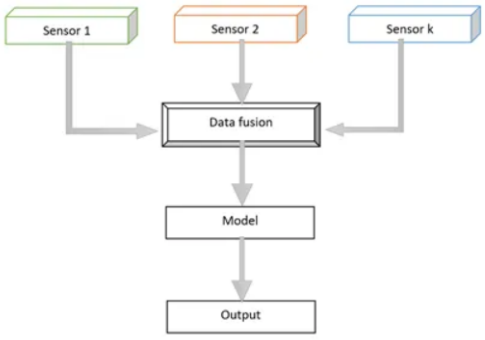


Figura 4.20: Early fusion.

Esta técnica presenta dos desventajas. La primera es que se deducirá una gran cantidad de datos de las modalidades para hacer una base común antes de la fusión. Una vez que los datos tienen matrices comunes, se analizan mediante un algoritmo de aprendizaje automático. La otra desventaja de este método es la sincronización de las marcas de tiempo de las distintas modalidades. Una forma habitual de superar este inconveniente es recoger los datos o señales a una frecuencia de muestreo común. Martínez et al. [10] proponen otras soluciones atenuantes, como la formación, la agrupación y la fusión por convolución. Estos métodos propuestos se lograron mediante la fusión de eventos discretos secuenciales con datos continuos.

4.8.2. Late fusion

La fusión tardía (también conocida como fusión a nivel de decisión) utiliza fuentes de datos independientes, seguidas de la fusión en una fase de toma de decisiones (como se puede ver en la siguiente figura). Esta técnica es mucho más sencilla que el método de fusión temprana, sobre todo cuando las fuentes de datos son muy distintas entre sí en cuanto a frecuencia de muestreo, dimensionalidad de los datos y unidad de medida. La fusión tardía suele dar mejores resultados porque los errores de varios modelos se tratan de forma independiente, por lo que los errores no están correlacionados. Sin embargo, Ramachandram et al. [11] afirman que no hay pruebas concluyentes de que la fusión tardía ofrezca mejores resultados que la fusión temprana. Aun así, muchos investigadores utilizan esta técnica para analizar problemas de datos multimodales.

Existen distintas técnicas para determinar la manera óptima de combinar cada uno de los modelos entrenados de forma independiente. Algunas de esas técnicas son: Bayes, la fusión máxima y la fusión media.

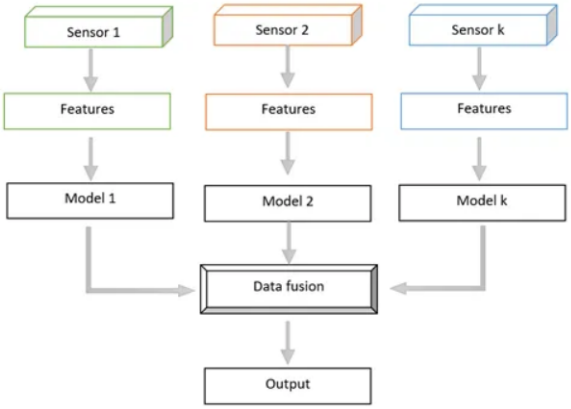


Figura 4.21: Late fusion.

4.8.3. Intermediate fusion

La arquitectura de la fusión intermedia se construye sobre la base de la red neuronal profunda. Este método es el más flexible, ya que permite la fusión de datos en diferentes etapas del entrenamiento del modelo.

La fusión intermedia transforma los datos de entrada en un nivel superior de representación (características) a través de múltiples capas. Cada capa individual opera funciones lineales y no lineales que transforman la escala, la inclinación y la oscilación de los datos de entrada y proporciona una nueva representación de los datos de entrada originales. Las características pueden aprenderse a partir de diferentes tipos de capas, incluyendo: Convolución 2D, Convolución 3D y totalmente conectadas. La capa en la que se fusionan las características de las distintas modalidades se denomina capa de fusión o capa de representación compartida.

Las distintas modalidades se pueden fusionar simultáneamente en una única capa de representación compartida, o se pueden fusionar gradualmente utilizando una o varias modalidades a la vez (como en la siguiente figura). Aunque es posible fusionar características o pesos de varias modalidades en una sola capa, puede dar lugar a un ajuste excesivo del modelo o a que la red no aprenda la relación entre cada modalidad.

En la siguiente figura se puede observar la arquitectura de este tipo de fusión:

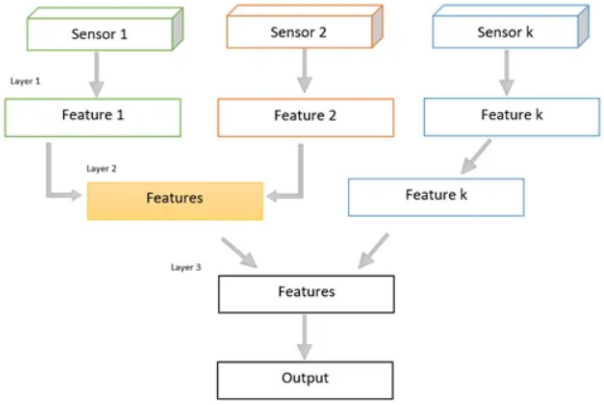


Figura 4.22: Intermediate fusion.

5. CAPÍTULO

Desarrollo del proyecto

En este capítulo se explica todo necesario para comprender la implementación del proyecto. La primera sección contiene la descripción de las bases de datos utilizadas y gráficas de la distribución de las clases. En la segunda sección, se describe el procesamiento de los datos en imagen, vídeo y audio. En la tercera sección se exponen las distintas técnicas de extracción de características del audio que se han utilizado y la representación del audio en cada una de ellas. En el cuarto capítulo se explica cómo se ha implementado cada modelo en las distintas modalidades. En el quinto capítulo de presentan las tablas con los resultados obtenidos. Finalmente, en el último capítulo se hace una comparación entre los mejores resultados obtenidos en este proyecto y los mejores resultados que han logrado los investigadores.

5.1. Bases de datos utilizadas

En esta sección se explicarán las distintas bases de datos utilizadas durante la realización del proyecto.

En la siguiente tabla se puede observar las bases de datos que se han utilizado para las distintas modalidades.

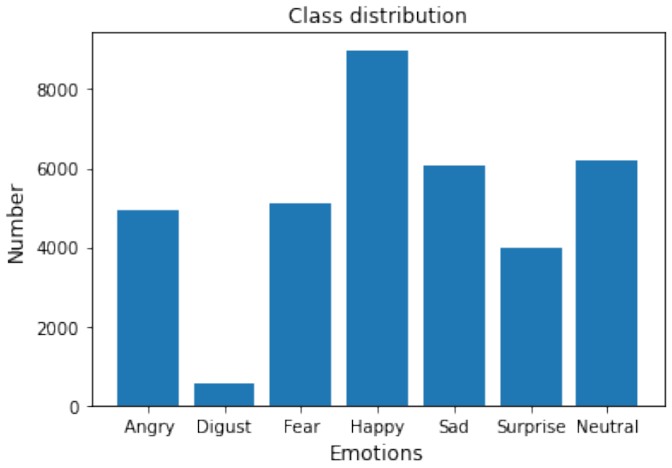
Base de datos	Modalidad
FER-2013	Imagen
RAVDESS	Vídeo y audio
Crema-D	Audio
SAVEE	Audio
TESS	Audio

Tabla 5.1: Bases de datos utilizadas.

5.1.1. FER-2013

Los datos de FER-2013 [1](Facial Emotion Recognition) consisten en imágenes de rostros en escala de grises de 48x48 píxeles. Las caras se han registrado automáticamente de modo que estén más o menos centradas y ocupen aproximadamente el mismo espacio en cada imagen.

El conjunto de datos está formado por 28709 imágenes etiquetadas en el conjunto de entrenamiento y 7178 imágenes etiquetadas en el conjunto de test. Cada imagen de FER-2013 está clasificada como una de siete emociones (0: enfado, 1: disgusto, 2: miedo, 3: felicidad, 4: tristeza, 5: sorpresa, 6: neutral). Para obtener este conjunto de datos, se hizo una búsqueda de las imágenes de las emociones y de sus sinónimos en Google y se recopilaron los resultados. Los datos están almacenados en un .csv.



(a) Gráfica de las emociones

	emotion	number
0	Angry	4953
1	Digust	547
2	Fear	5121
3	Happy	8989
4	Sad	6077
5	Surprise	4002
6	Neutral	6198

(b) Recuento de las emociones

Figura 5.1: Distribución de las emociones en FER-2013.

[1]https://www.kaggle.com/datasets/deadskull7/fer2013

5.1.2. RAVDESS

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) [5] contiene 7356 archivos (tamaño total: 24,8GB). La base de datos contiene 24 actores profesionales (12 mujeres y 12 hombres) que interpretan dos frases de léxico similar con acento neutro norteamericano. Cada expresión se produce en dos niveles de intensidad emocional(normal y fuerte), con una expresión neutra adicional. Hay tres formatos disponibles de modalidad: sólo audio (16bit, 48kHz .wav), audio-vídeo (720p H.264, AAC 48kHz, .mp4) y sólo vídeo, sin sonido.

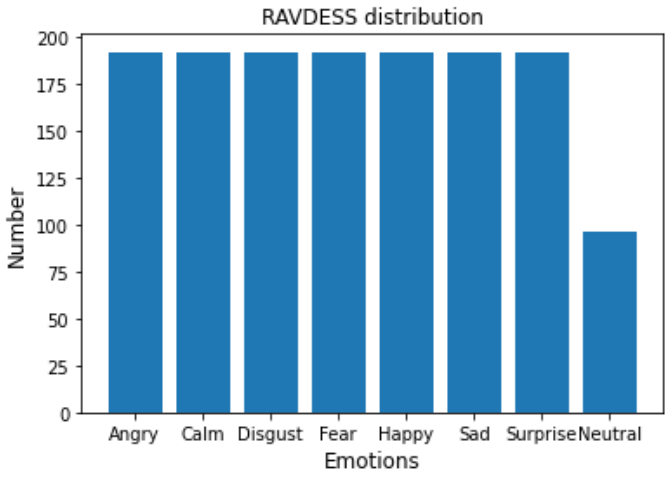
Para el desarrollo de este proyecto, se han utilizado las modalidades solo vídeo sin sonido (para el reconocimiento de emociones en vídeo) y sólo audio (reconocimiento de emociones en audio). Cada uno de los archivos de RAVDESS tiene un nombre de archivo único. El nombre del archivo consta de un identificador numérico de 7 partes (por ejemplo, 02-01-06-01-02-01-12.mp4). Estos identificadores definen las características del módulo:

- Modalidad (01: audio-vídeo, 02: sólo vídeo, 03: sólo audio).
- Canal vocal (01: discurso, 02: canción).
- Emoción (01: neutro, 02: calmado, 03: alegre, 04: triste, 05: enfadado, 06: temeroso, 07: disgustado, 08: sorprendido).
- Intensidad emocional (01: normal, 02: fuerte).
- Enunciado (01: "Kids are talking by the door", 02: "Dogs are sitting by the door").
- Repetición (01: 1º repetición, 02: 2º repetición).
- Actor (01 a 24. Los actores impares son hombres y los pares son mujeres).

Ejemplo de nombre de un fichero RAVDESS: 02-01-06-01-02-01-12.mp4.

1. Sólo vídeo (02)
2. Discurso (01)
3. Temeroso (06)

4. Intensidad normal (01)
5. "Dogs are sitting by the door"(02)
6. 1º repetición (01)
7. Doceavo actor (12)
8. Mujer (ya que el número de identificación del actor es par)



Angry	192
Calm	192
Disgust	192
Fear	192
Happy	192
Sad	192
Surprise	192
Neutral	96

(a) Gráfica de las emociones **(b)** Recuento de las emociones

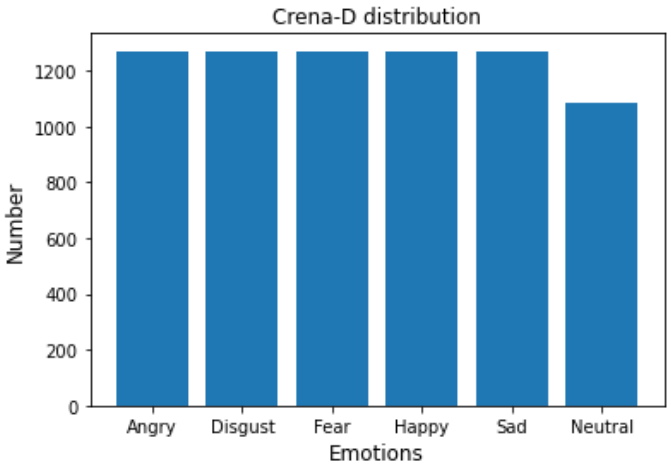
Figura 5.2: Distribución de las emociones en RAVDESS.

5.1.3. Crema-D

Crowd-sourced Emotional Mutimodal Actors Dataset (Crema-D)[2] es un conjunto de datos de 7.442 clips originales de 91 actores (48 actores masculinos y 43 femeninos) con edades comprendidas entre los 20 y los 74 años y procedentes de una variedad de razas y etnias (afroamericana, asiática, caucásica, hispana y no especificada). Cada actor dice 12 frases. Las frases se presentan utilizando una de las seis emociones diferentes (ira, disgusto, miedo, alegría, neutralidad y tristeza) y cuatro niveles de emoción diferentes (bajo, medio, alto y sin especificar). Cada archivo de Crema-D consta de un identificador único de 4 partes (por ejemplo, 1001_IWW_NEU_XX.wav). Estos identificadores definen las características del módulo:

[2]https://www.kaggle.com/datasets/ejlok1/cremad

- ID del actor.
- Frase que dice el actor, entre paréntesis aparece el acrónimo de tres letras utilizado en la segunda parte del nombre del archivo:
  - It's eleven o'clock (IEO).
  - That is exactly what happened (TIE).
  - I'm on my way to the meeting (IOM).
  - I wonder what this is about (IWW).
  - The airplane is almost full (TAI).
  - Maybe tomorrow it will be cold (MTI).
  - I would like a new alarm clock (IWL)
  - I think I have a doctor's appointment (ITH).
  - Don't forget a jacket (DFA).
  - I think I've seen this before (ITS).
  - The surface is slick (TSI).
  - We'll stop in a couple of minutes (WSI).
- Emoción, entre paréntesis aparece el acrónimo de tres letras utilizado en la tercera parte del nombre del archivo: ira (ANG), disgusto (DIS), miedo (FEA), alegría (HAP), neutro (NEU), triste (SAD).
- Nivel de emoción: bajo (LO), medio (MD), alto (HI), sin especificar (XX)



Angry	1271
Disgust	1271
Fear	1271
Happy	1271
Sad	1271
Neutral	1087

(a) Gráfica de las emociones **(b)** Recuento de las emociones

Figura 5.3: Distribución de las emociones en Drema-D.

5.1.4. SAVEE

La base de datos Surrey Audio-Visual Expressed Emotion (SAVEE)[3] se ha generado a partir de cuatro hablantes nativos ingleses de género masculino (identificados como DC, JE, JK y KL, estudiantes de posgrado e investigadores de la Universidad de Surrey con edades comprendidas entre los 27 y los 31 años. En esta base de datos se diferencian 6 diferentes emociones: ira, disgusto, miedo, alegría, tristeza y sorpresa. Así lo corroboran los estudios transculturales de Ekman y los estudios sobre el reconocimiento automático de emociones tienden a centrarse en el reconocimiento de éstas. Por cada emoción se distinguen 15 frases: 3 comunes, 2 específicas de cada emoción y 10 genéricas, diferentes para cada emoción y equilibradas fonéticamente. Las 3 frases comunes y las 12 frases específicas de cada emoción se grabaron como neutras para obtener 30 frases neutras. El resultado fue un total de 120 frases por hablante, por ejemplo:

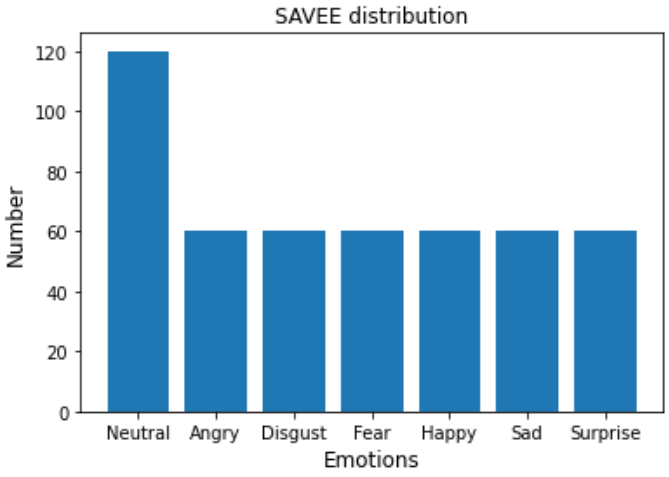
- **Común:** She had your dark suit in greasy wash water all year.
- **Ira:** Who authorized the unlimited expense account?
- **Disgusto:** Please take this dirty table cloth to the cleaners for me.
- **Miedo:** Call an ambulance for medical assistance.
- **Alegría:** Those musicians harmonize marvelously.
- **Tristeza:** The prospect of cutting back spending is an unpleasant one for any governor.
- **Sorpresa:** The carpet cleaners shampooed our oriental rug.
- **Neutro:** The best way to learn is to solve extra problems.

Cada archivo de SAVEE consta de un identificador único de 3 partes (por ejemplo, DC_a11.wav).

1. ID del orador (DC, JE, JK y KL).
2. Emoción (a: enfado, d: disgusto, f: miedo, h: alegría, n: neutro, sa: tristeza, su: sorpresa).

[3] https://www.kaggle.com/datasets/ejlok1/surrey-audiovisual-expressed-emotion-savee

3. Oración (01-30).



Neutral	120
Angry	60
Disgust	60
Fear	60
Happy	60
Sad	60
Surprise	60

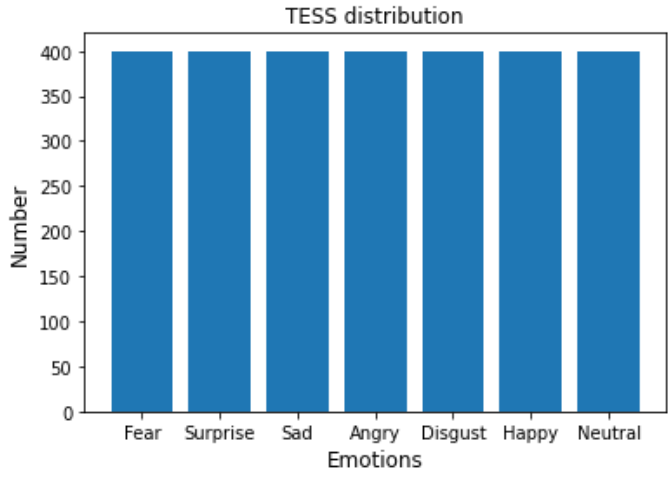
(a) Gráfica de las emociones **(b)** Recuento de las emociones

Figura 5.4: Distribución de las emociones en SAVEE.

5.1.5. TESS

En Toronto emotional speech set (TESS) [6] todas las oradoras son mujeres. Hay un conjunto de 200 palabras clave que fueron pronunciadas en la frase " say the word... " por dos actrices (de 26 y 64 años) y se hicieron grabaciones del conjunto representando cada una de las siete emociones (ira, disgusto, miedo, felicidad, sorpresa, tristeza y neutro). En total hay 2.800 archivos de audio. El conjunto de datos está organizado de forma que cada una de las dos actrices y sus emociones están contenidas en su propia carpeta. Y dentro de ésta, se encuentran los 200 archivos de audio. Los archivos de audio están en formato WAV. Cada archivo de TESS consta de un identificador único de 2 partes (por ejemplo, OAF_room_angry.wav).

- ID de la oradora (OAF, YAF).
- Palabra clave (en el ejemplo sería room).
- Emoción



Fear	400
Surprise	400
Sad	400
Angry	400
Disgust	400
Happy	400
Neutral	400

(a) Gráfica de las emociones **(b)** Recuento de las emociones

Figura 5.5: Distribución de las emociones en TESS.

5.2. Procesamiento de los datos

En esta sección se expondrán los pasos que se han seguido para procesar los datos en las distintas modalidades.

5.2.1. Imagen

El archivo fer2013.csv contiene tres columnas: emoción, píxeles y uso (conjunto de training o de test), aunque para el desarrollo del proyecto no se ha utilizado esta última columna. Para procesar los datos de las imágenes, se ha leído el archivo línea por línea de tal manera que se van añadiendo las emociones en un vector y los píxeles en otro.

Cabe mencionar que para implementación del modelo Inception v3 las imágenes se han redimensionado a 139x139 para evitar problemas con el tamaño de entrada del modelo pre-entrenado, (en lo otros modelos se mantienen los 48x48 píxeles).

Después, se han normalizado los valores de los píxeles. Para ello, se ha realizado una transformación de cada píxel: “valor/255”, de manera que siempre quede un valor entre 0 y 1 y, se actualizan los vectores mencionados anteriormente. Se divide el valor de cada píxel entre 255 porque los pixeles toman valores entre 0 y 255.

Posteriormente, ambos vectores se almacenan en dos archivos .npy para no repetir la misma operación cada vez que se quiera acceder a estos datos o entrenar un nuevo modelo.

El siguiente paso (común para todas las modalidades) que se ha realizado es aplicar stratified k fold cross validation (k=5) en el conjunto de datos. De esta manera, los datos se dividen en dos conjuntos: conjunto de training y conjunto de test, como ya se ha explicado en la sección 4.7.

Una vez se ha realizado la división de los conjuntos, los datos se introducen en las redes neuronales para entrenarlas y obtener los resultados. Como se ha utilizado stratified k fold cross validation, el entrenamiento de cada red neuronal y el cálculo de las distintas métricas, se han realizado dentro de un bucle que recorre todos los subconjuntos. Por lo tanto, se hace un entrenamiento por cada uno de los cinco subconjuntos y las métricas obtenidas son la media de cada subconjunto.

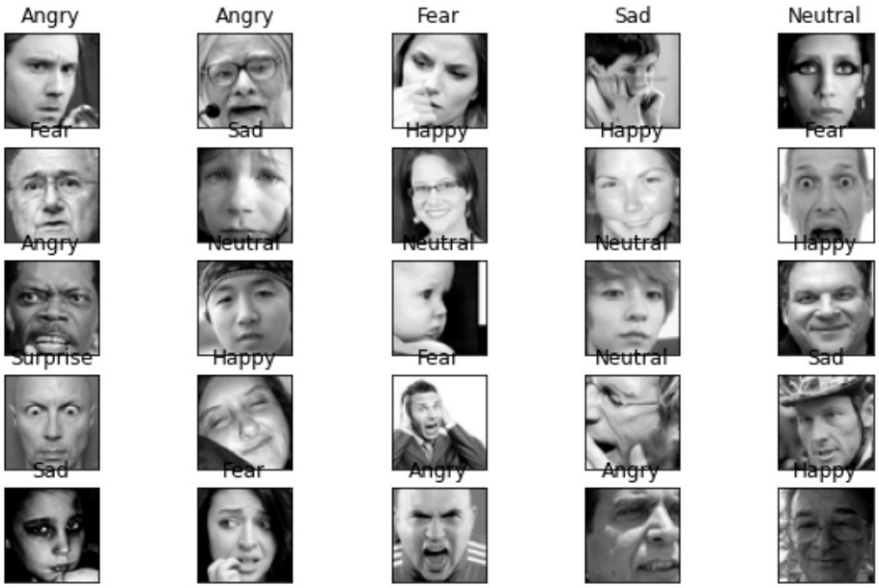


Figura 5.6: Algunas imágenes de FER-2013.

En la imagen 5.6 se pueden observar ejemplos de los emociones de FER-2013.

5.2.2. Vídeo

La base de datos RAVDESS contiene 1438 vídeos, cada uno de ellos dura 3 segundos. Para reconocer las emociones en esta modalidad, se han dividido los vídeos en fotogramas a 30fps. Es decir, a partir de un vídeo de 3 segundos, se extraen 90 imágenes, 129.420 imágenes en total. Los fotogramas extraídos del vídeo son de 438x438 píxeles.

Para extraer los fotogramas de los vídeos se ha utilizado la librería OpenCV. El vídeo se lee mediante la función "VideoCapture" y se captura el fotograma con la función "read()" como se puede observar en el siguiente fragmento de código:

```
videocap = cv2.VideoCapture(os.path.join(*(videopath, video_list[i])))
success,frame = videocap.read()
```

Cada vez que se extrae un fotograma del vídeo, hay que detectar la cara de la persona, para ello se ha utilizado la función "detectMultiScale()" del clasificador "Haar Cascade"[4] a escala 1.1 y minNeighbors=4 (parámetro que especifica cuántos vecinos debe tener cada rectángulo candidato para conservarlo). Después, cada fotograma del vídeo se almacena en la carpeta de clase a la que pertenece.

Hay distintas técnicas para reconocer caras en imágenes, en este proyecto se ha utilizado el clasificador Haar Cascade o también conocido como algoritmo de Viola-Jones[5]. Se ha decidido utilizar este clasificador porque es de los clásicos (que no el mejor). Para que este algoritmo funcione es necesario entrenarlo mostrándole rostros, y otras cosas que no sean rostros, para que aprenda a distinguirlos. El proceso puede ser tedioso, ya que se necesitan una gran cantidad de imágenes. Por suerte, ya hay clasificadores pre-entrenados para reconocer caras (y otras muchas cosas), que están almacenas en formato .xml y que podemos cargar desde OpenCV. El que se ha utilizado se llama haarcascade_frontalface_default.xml[6]. Está entrenado con rostros mirando al frente, como sucede en las imágenes que se han utilizado en el proyecto.

Después de tener los 8 directorios diferentes (uno por cada emoción) con sus correspondientes fotogramas recorremos los directorios para obtener todas las imágenes. Estas imágenes se re-dimensionan las a 75x75 píxeles (para los modelos Inception v3, VGG16 y VGG19) y a 48x48 píxeles para los demás modelos. Las imágenen se han re-dimensionado a 75x75 píxeles para evitar posibles problemas con los modelos pre-entrenados y su formato de entrada.

[4] https://docs.opencv.org/3.4/db/d2"/tutorial_cascade_classifier.html

[5] https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999

[6] https://raw.githubusercontent.com/opencv/opencv/master/data/haarcascades/haarcascade_frontalface_default.xml

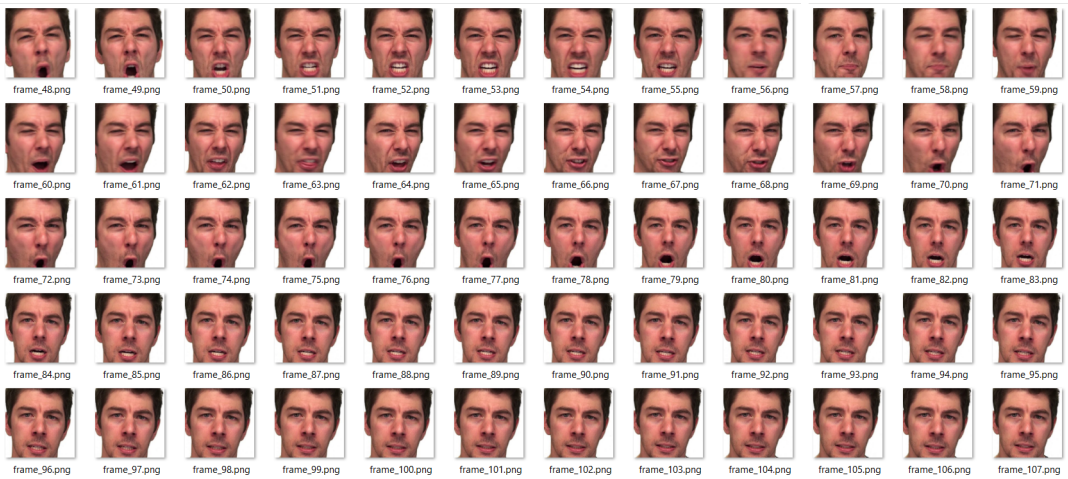


Figura 5.7: Algunos fotogramas extraídos de RAVDESS.

5.2.3. Audio

RAVDESS

Los archivos de audio de la base de datos RAVDESS están distribuidos en 24 carpetas, una carpeta por cada actor. Por cada carpeta, se lee el nombre de cada archivo y se añade en un DataFrame[7] el identificador del audio, su correspondiente emoción, y su ruta. Para poder introducir los datos del audio en las redes neuronales es necesario extraer sus características, explicado en la sección 5.3.

Crema-D

Todos los archivos de audio de la base de datos Crema-D están almacenados en la misma carpeta. Por cada audio, se lee el nombre del archivo y en un vector se añade la ruta del archivo y en otro su correspondiente emoción. Después, estos datos se almacenan en un DataFrame de dos columnas (emoción y ruta del archivo).

SAVEE

Todos los archivos de audio de la base de datos SAVEE están almacenados en el mismo directorio. Por cada audio, se lee el nombre del archivo y en un vector se añade la ruta del

[7] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

archivo y en otro su correspondiente emoción. Después, estos datos se almacenan en un DataFrame de dos columnas (emoción y ruta del archivo).

TESS

Los archivos de audio de la base de datos TESS también están almacenados en la misma carpeta. Por lo tanto el procedimiento es el mismo, se lee el nombre de cada archivo y la ruta del archivo se añade a un vector y su correspondiente emoción a otro. Después, estos datos se almacenan en un DataFrame de dos columnas (emoción y ruta del archivo).

5.3. Extracción de características del audio

Una vez se han obtenido los DataFrames, lo siguientes es utilizar las distintas ténicas para extraer características útiles del audio para nuestro clasificador. La extracción de características de audio es un paso necesario en el procesamiento de señales de audio, que es un subcampo del procesamiento de señales. Se ocupa del procesamiento o la manipulación de señales de audio. Elimina el ruido no deseado y equilibra los rangos de tiempo-frecuencia mediante la conversión de señales digitales y analógicas.

A la hora de utilizar redes neuronales para la clasificación de audio o texto la opción más obvia quizás sea utilizar redes recurrentes, ya que están pensadas para trabajar con secuencias con una relación temporal. Sin embargo, este modelo tiene el inconveniente de que son bastante lentas, lo cual puede ser un problema tanto para su entrenamiento, es por ello que se ha decidido utilizar tanto redes neuronales convolucionales como redes neuronales recurrentes para la clasificación del audio.

Al utilizar redes neuronales convolucionales, es necesario representar los archivos de audio audio como imágenes. Para ello, se ha decidido extraer las siguientes técnicas de extracción de características del audio: MFCC[8](Mel Frequency Cepstral Coefficients), Chroma [7], Mel Spectogram y STFT.

Mediante la librería librosa se puede obtener tanto la representación de las distintas características como la representación del audio en forma de una onda, donde el eje x es el tiempo en segundos el eje y corresponde a la amplitud del audio.

[8]https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/

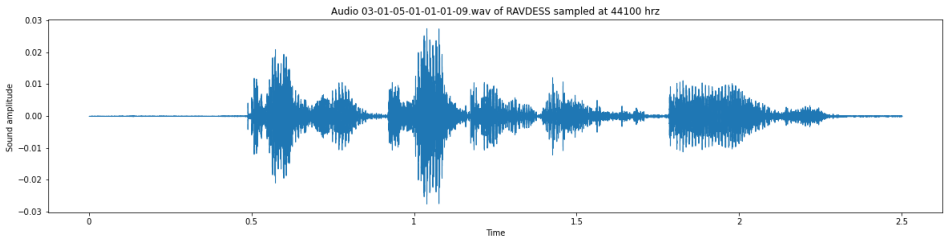


Figura 5.8: Representación del audio en forma de onda.

En las siguientes imágenes representadas por los coeficientes MFCC, se puede observar la comparación entre la misma frase dicha por un hombre y por una mujer. En la primera imagen se compara una frase que representa enfado mientras que en la segunda imagen se compara una frase que representa alegría.

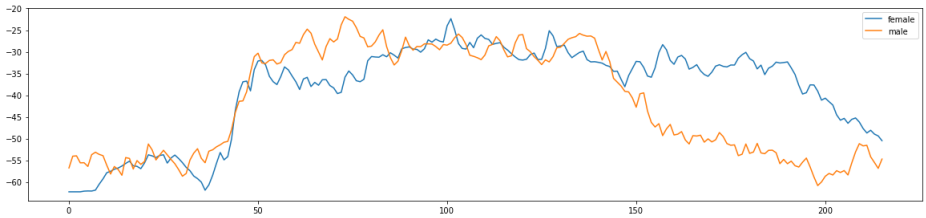


Figura 5.9: Comparación de una frase que representa enfado.

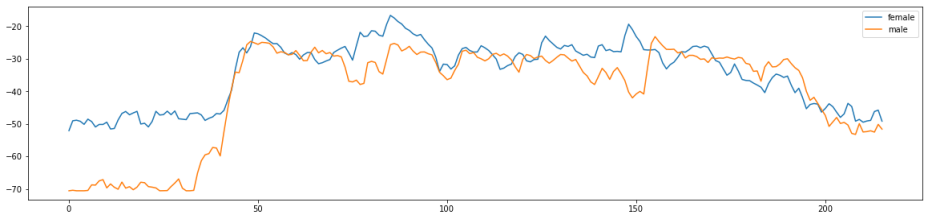


Figura 5.10: Comparación de una frase que representa alegría.

Para la misma frase que se pronuncia, hay una clara diferencia entre el hombre y la mujer ya que las mujeres tienen un tono más agudo. Por esta razón, en algunos casos resulta más sencillo detectar las emociones en las mujeres que en los hombres.

5.3.1. MFCC

Los MFCC son coeficientes para la representación del habla basados en la percepción auditiva humana. Estos coeficientes surgen de la necesidad de extraer las características de

las componentes de una señal de audio que sean adecuadas para identificar contenido relevante y, de la misma manera, no tener en cuenta la información menos valiosa, así como el ruido de fondo, el volumen, tono, etc. Los MFCCs son una de las características más utilizadas en el reconocimiento del audio. Los MFCC se basa en escala Mel que relaciona la frecuencia percibida de un tono con la frecuencia real medida. Escala la frecuencia para ajustarse a lo que puede escuchar el ser humano.

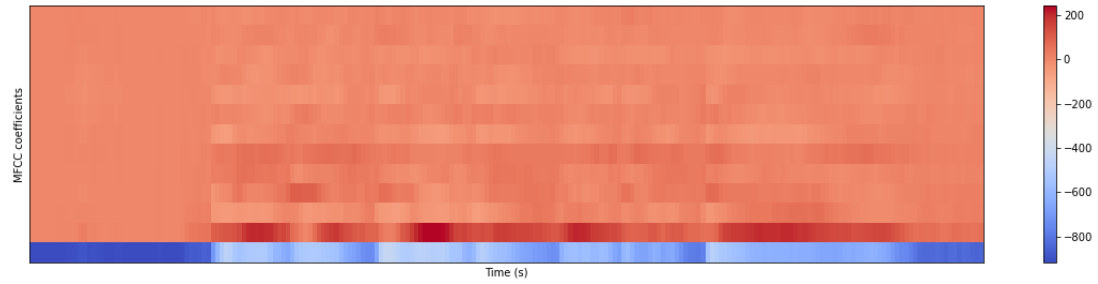


Figura 5.11: Ejemplo de las características MFCC de un archivo de audio.

5.3.2. Chroma

Un vector de Chroma es un vector de características, generalmente de 12 elementos que indica la energía del tono de cada clase en la señal en una escala cromática estándar. Esta característica se utiliza mucho en la música ya que son resistentes a los cambios de timbre e instrumentación.

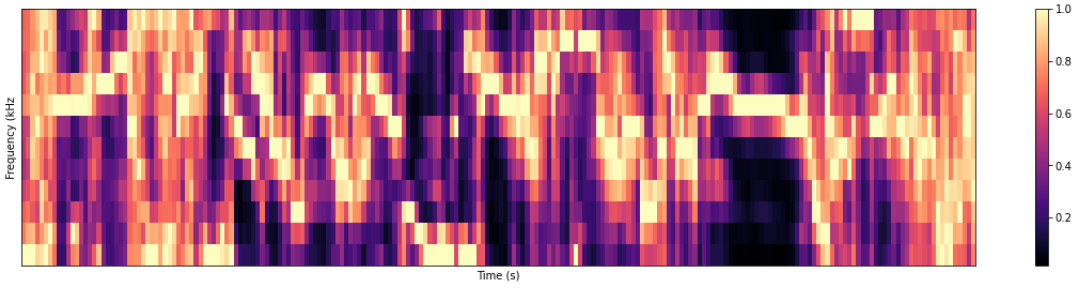


Figura 5.12: Ejemplo de las características Chroma de un archivo de audio.

5.3.3. Mel Spectogram

Un espectrograma mel es un espectrograma en el que las frecuencias se convierten a la escala mel. La escala mel fue propuesta Stevens, Volkman y Newmann en 1937, es una escala perceptiva de tonos que los oyentes consideran iguales en distancia entre sí.

El punto de referencia entre esta escala y la medición de frecuencia normal se define asignando un tono de percepción de 1000 mels a un tono de 1000 Hz , 40 dB por encima del umbral del oyente.

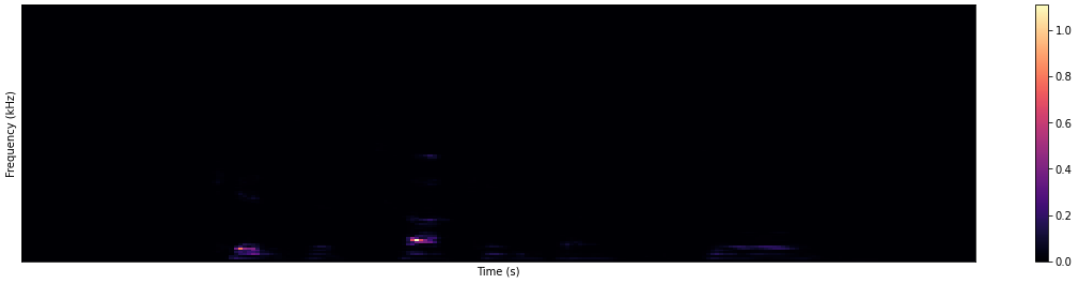


Figura 5.13: Ejemplo del Mel Spectogram de un archivo de audio.

5.3.4. STFT

Short-Time Fourier Transform (STFT) es una técnica de análisis de señal utilizada en procesamiento de audio y señales de tiempo-frecuencia. STFT divide una señal de audio en segmentos cortos o ventanas y luego aplica la transformada de Fourier a cada ventana para producir una representación de la frecuencia de la señal en ese momento temporal específico. Esto permite a los usuarios visualizar y analizar la información de frecuencia de una señal a lo largo del tiempo. La STFT es ampliamente utilizada en aplicaciones como la eliminación de ruido, la separación de señales y la identificación de componentes específicos de la señal.

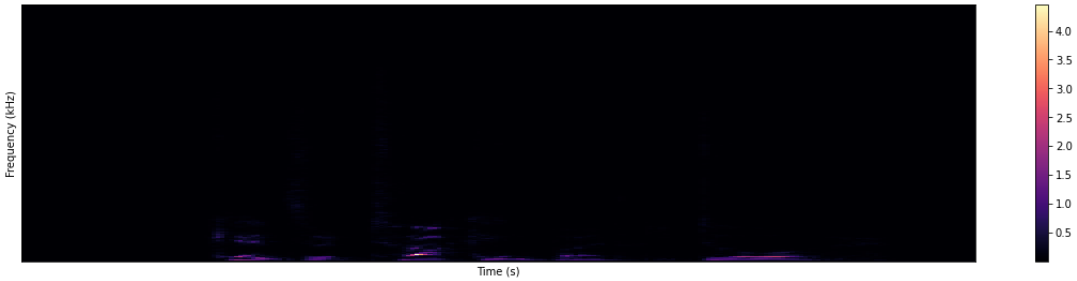


Figura 5.14: Ejemplo del STFT de un archivo de audio.

5.4. Redes neuronales implementadas

En esta sección se explicarán las distintas implementaciones que se han realizado durante el proyecto. Se ha decidido no utilizar el aumento de datos e implementar redes neuronales recurrentes, residuales y distintas variaciones de las redes neuronales convolucionales. Aunque en los artículos que recogen los trabajos de otros investigadores se expone que se han utilizado fusiones de distintos tipos redes neuronales y, aumento de datos, se ha optado implementar redes neuronales más sencillas y no utilizar aumento de datos. La razón es simple, por un lado, antes de comenzar con el proyecto no se había trabajado con redes neuronales, por lo que se ha decidido elegir modelos más sencillos. Por otro lado, en los distintos concursos que hay en Kaggle para el reconocimiento de las emociones, se ha observado que distintos concursantes utilizaban estos modelos y en algunos casos, los resultados eran buenos.

Por cada red neuronal implementada se han ido modificando los hiperparámetros para hacer pruebas y ver cuál es la configuración que nos puede ofrecer mejores resultados. A continuación se explicarán las redes neuronales implementadas con sus hiperparámetros, se exponen los que han dado mejores resultados entre todas sus pruebas realizadas.

5.4.1. AlexNet

En la modalidad imagen:

- Se crea el modelo secuencial.
- Se añade al modelo una capa **Convolutional2D** con 64 filtros, strides (5,5) y se indica el tamaño de entrada. En este caso el tamaño de entrada es (48,48,1), ya que son imágenes de 48x48 píxeles en escala de grises.
- Capa de **activación** relu.
- Capa **Convolutional2D** con 64 filtros y strides (5,5).
- Capa de **activación** relu.
- Capa **MaxPooling 2D** (3x3 píxeles).
- Capa **Convolutional2D** con 64 filtros y strides (3,3)

- Capa de **activación** relu.
- Capa **Convolutional2D** con 64 filtros y strides (3,3)
- Capa de **activación** relu.
- Capa **MaxPooling 2D** (3x3 píxeles).
- Capa **Flatten**, se usa para "aplanar " la entrada.
- Capa **Dense** con 512 unidades.
- Capa de **activación** relu.
- Capa **Dropout** 0.5, sirve para reducir el sobreajuste.
- Capa **Dense** de 7 unidades, capa de salida de tantas unidades como clases.
- Capa de **activación** softmax. La función softmax se utiliza como capa final de las redes neuronales.

En las modalidades audio y vídeo:

```
def build_AlexNet_model():
    model = keras.models.Sequential([
        keras.layers.Conv2D(filters=96, kernel_size=(11,11),
        strides=(4,4),
        activation='relu', input_shape=(48,48,3)),
        keras.layers.BatchNormalization(),
        keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
        keras.layers.Conv2D(filters=256, kernel_size=(5,5),
        strides=(1,1),
        activation='relu', padding="same"),
        keras.layers.BatchNormalization(),
        keras.layers.MaxPool2D(pool_size=(3,3), strides=(2,2)),
        keras.layers.Conv2D(filters=384, kernel_size=(3,3),
        strides=(1,1),
        activation='relu', padding="same"),
        keras.layers.BatchNormalization(),
```

```
        keras.layers.Conv2D(filters=384, kernel_size=(3,3),
        strides=(1,1),
        activation='relu', padding="same"),
        keras.layers.BatchNormalization(),
        keras.layers.Conv2D(filters=256, kernel_size=(3,3),
        strides=(1,1),
        activation='relu', padding="same"),
        keras.layers.BatchNormalization(),
        keras.layers.MaxPool2D(pool_size=(3,3),
        strides=(2,2), padding="same"),
        keras.layers.Flatten(),
        keras.layers.Dense(4096, activation='relu'),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(4096, activation='relu'),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(numclasses, activation='softmax')
    ])
    return model
```

Tanto en la modalidad de vídeo como en la modalidad de audio, la red neuronal se ha implementado de la misma forma. Lo único que varía es el tamaño de entrada y el número de clases de la última capa. En el caso del audio serán tres parámetros, los dos primeros comunes y el último variará en función de la característica de extracción del audio seleccionada. En el caso del vídeo, el tamaño de entrada es (48,48,3), imágenes de 48x48 píxeles en RGB. El número de clases dependerá de las distintas emociones que se hayan clasificado en la base de datos.

5.4.2. VGG16

En la modalidad imagen:

```
def build_VGG16_model():
    base_model = tf.keras.applications.VGG16(input_shape=(48,48,3),
    include_top=False,weights="imagenet")
    for layer in base_model.layers[:-4]:
```

```
    layer.trainable=False
model = Sequential([
  tf.keras.layers.Lambda(tf.image.grayscale_to_rgb),
  base_model])
model.add(Dropout(0.5))
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(32,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(32,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(32,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dense(7,activation='softmax'))
return model
```

Primero se crea el modelo base a partir de la red pre-entrenada con los pesos de la base de datos ImageNet. Después se congelan las capas para evitar que los pesos se modifiquen. Algo muy importante a tener en cuenta es que se hace uso de la función grayscale_to_rgb, esta función cambia el canal de las imágenes, es decir, las pasa de RGB a escala de grises. Las imágenes de ImageNet están en RGB mientras que las imágenes que tenemos en FER-2013 no, si no se utilizase esta función habría muchos problemas en cuanto al tamaño de entrada de la red neuronal y no se podría probar este modelo. Se utiliza un la capa Dropout con una proporción de 0.5 y activación relu excepto en la última capa que se utiliza la activación softmax.

En la modalidad de vídeo, se ha implementado la red neuronal de la misma forma que para la imagen. La única diferencia es, por un lado, que no se necesita pasar las imágenes a escala de grises (porque los fotogramas extraídos de la base de datos RAVDESS están en RGB). Por otro lado, el tamaño de entrada que es (75,75,3). Finalmente, el número de clases dependerá del número de emociones que haya en cada base de datos (como ya se ha explicado anteriormente).

5.4.3. VGG19

En la modalidad imagen:

```
def build_VGG19_model():
    base_model = tf.keras.applications.VGG19(input_shape=(48,48,3),
    include_top=False,weights="imagenet")
    # Freezing Layers
    for layer in base_model.layers[:-4]:
        layer.trainable=False
    model = Sequential([
      tf.keras.layers.Lambda(tf.image.grayscale_to_rgb),
      base_model])
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(BatchNormalization())
    model.add(Dense(32,kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(32,kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(32,kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dense(7,activation='softmax'))
    return model
```

En la modalidad de vídeo, se ha implementado la red neuronal de la misma forma que para la imagen. Las diferencias son las mismas que se han comentado anteriormente, no se necesita pasar las imágenes de RGB a escala de grises, el tamaño de entrada es diferente, en este caso (75,75,3) y el número de clases también difiere.

5.4.4. Inception v3

En la modalidad de imagen:

```
def build_Inceptionv3_model():
    base_model = InceptionV3(include_top = False, weights= 'imagenet',
    input_shape = (139, 139, 3))
    # Freezing Layers
    for layer in base_model.layers[:-4]:
        layer.trainable=False
    model = Sequential([
      tf.keras.layers.Lambda(tf.image.grayscale_to_rgb),
      base_model])
    model.add(GlobalAveragePooling2D())
    model.add(Dense(7,activation='softmax'))
    return model
```

Primero, se crea el modelo base con los pesos de la red neuronal pre-entrenada. En este caso, el tamaño de entrada es de (139,139,3), Inception v3 no admite imágenes de menos de 75x75 píxeles por lo que se ha decidido re-dimensionar las imágenes de FER-2013 a 139x139 píxeles. Después se pasan las imágenes de RGB a escala de frises. Finalmente, se añade la capa **GlobalAveragePooling** y la capa **Dense** de salida.

En la modalidad de vídeo, esta red se ha implementado de la misma forma salvo que el tamaño de entrada es de 75x75 píxeles y una vez más, no es necesario hacer el cambio en el canal de las imágenes.

5.4.5. CNN

En las tres modalidades:

- Se crea el modelo secuencial.
- Se añade la capa de convolución **Conv2D** con 32 filtros 3x3 píxeles, activación relu. Los tamaños de entrada son los siguientes: (48,48,1) para las imágenes y (48,48,3) para el vídeo. Para el vídeo el tamaño de entrada variará según la base de datos y la característica seleccionada.

- Se añade la capa **Pooling** (2,2).
- Se añade la capa **Flatten** (fully connected).
- Capa **Dense** con 100 unidades.
- Capa de **Dense** de salida, con tantas unidades como clases hay en la base de datos y activación softmax.

5.4.6. CNN LSTM

Combinación entre una CNN y una LSTM. La razón de haber realizado esta combinación es que las redes LSTM se utilizan especialmente en el reconocimiento del audio y no en las imágenes porque no aportan resultados tan buenos. Como con la red neuronal convolucional se obtienen buenos resultados en ambas modalidades se ha decidido hacer una combinación entre ambas para ver si se consigue mejorar los resultados. En las tres modalidades se ha implementado de la siguiente forma:

```
def build_CNN_LSTM_model():
    model = Sequential()
    model.add(Conv2D(16, (2,2), padding = 'same',
    input_shape=(48, 48, 1)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(32, (2,2), padding = 'same'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.2))
    model.add(TimeDistributed(Flatten()))
    model.add(LSTM(100))
    model.add(Dense(numclasses, activation='softmax'))
    return model
```

Es importante añadir la capa TimeDistributed antes de Flatten porque sino se pueden producir errores no deseados al combinar una CNN con una LSTM. Lo que varía entre

la implementación de unas modalidades a otras es el tamaño de entrada y el número de unidades en la última capa densa.

5.4.7. LSTM

En las modalidades de imagen y video:

```
def build_LSTM_model():
    model = Sequential()
    model.add(TimeDistributed(Flatten()))
    model.add(LSTM(100))
    model.add(Dense(numClass, activation='softmax'))
    return model
```

En el audio:

```
def build_LSTM_model():
    input_shape=get_input_shape(feature)
    model = tf.keras.Sequential()

    model.add(LSTM(64, input_shape=input_shape, return_sequences=True))
    model.add(LSTM(32))

    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.3))

    model.add(Dense(numclasses, activation='softmax'))

    return model
```

Para obtener el tamaño de entrada en esta modalidad, se llama a una función que devuelve el tamaño correspondiente a la base de datos y a la característica seleccionada. Primero, se crea el modelo secuencial. Después, se añade la capa **LSTM** con 64 unidades, el tamaño de entrada correspondiente y return_sequences=True para que la segunda capa LSTM tenga una entrada de secuencia tridimensional. Lo siguiente es añadir otra capa **LSTM**

con 32 unidades. Después, se añade una capa **Dense** con 32 unidades y función de activación relu. Siguiéndole, se añade una capa Dropout 0,3 y por último se añade la capa de salida con tantas unidades como clases hay y activación softmax.

5.4.8. DCNN

En imagen y vídeo (salvo el tamaño de entrada que son diferentes, (48,48,1) en imagen y (48,,48,3) vídeo:

```
def build_DCNN_model():
    model = Sequential()
    model.add(Conv2D(filters = 64, kernel_size = (3,3),
    activation = 'elu',
    kernel_initializer='he_normal', padding='same',
    input_shape = (48,48,1)))
    model.add(BatchNormalization())
    model.add(Conv2D(filters = 64, kernel_size = (3,3),
    activation = 'elu',
    kernel_initializer='he_normal',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2,2),padding='same'))
    model.add(Dropout(0.3))


    model.add(Conv2D(filters = 128, kernel_size = (3,3),
    activation = 'elu',
    kernel_initializer='he_normal',padding='same'))
    model.add(BatchNormalization())
    model.add(Conv2D(filters = 128, kernel_size = (3,3),
    activation = 'elu',
    kernel_initializer='he_normal',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2,2),padding='same'))
    model.add(Dropout(0.4))
```

```
model.add(Conv2D(filters = 256, kernel_size = (3,3),
activation = 'elu',
kernel_initializer='he_normal',padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(filters = 256, kernel_size = (3,3),
activation = 'elu',
kernel_initializer='he_normal',padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2),padding='same'))
model.add(Dropout(0.5))


model.add(Flatten())

model.add(Dense(units = 1024, activation = 'relu',
kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(units = 512, activation = 'relu',
kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(units = 256, activation = 'relu',
 kernel_initializer='he_uniform'))
model.add(BatchNormalization())

model.add(Dense(units = numclasses, activation = 'softmax'))
return model
```

5.4.9. ResNet50

Se ha implementado de la misma forma tanto en imagen como en vídeo. Primero se crea el bloque de identidad, este bloque está compuesto por tres componentes distintos. En los primeros dos componentes, se añade una capa convolucional, seguida de una capa de normalización y una capa de activación relu. En el siguiente componente se añade una capa de convolución seguida de una capa de normalización. El tercer componente se utiliza pata hacer un 'shortcut'. ResNet está equipada con shortcuts, que omiten capas en el paso hacia delante de una entrada. El último paso es añadir el shortcut a la ruta principal, y pasarlo a través de una activación relu.

Después, se crea el bloque convolucional. Este bloque también está formado por tres componentes distintos. Los dos primeros están formados por una capa convolucional seguida de una capa de normalización y una activación relu. El siguiente componente está formado por una capa de convolución y una capa de normalización. El último componente es utilizado para el shortcut. Para finalizar, se añade el shortcut a la ruta principal y se pasa a través de una activación relu.

Para crear el modelo ResNet es necesario mezclar ambos bloques.

En la primera fase, añadimos una capa convolucional de 8 filtros, tamaño de kernel (3,3) y strides (1,1). Después añadimos una capa de normalización seguida de una capa de activación relu.

En la segunda fase, añadimos el bloque convolucional con los filtros (32, 32, 128) seguido de dos bloque de identidad, ambos con los filtros (32, 32, 128).

En la tercera fase, añadimos el bloque convolucional con los filtros (64,64,256) seguido de tres bloques de identidad con los mismos filtros.

En la cuarta fase, añadimos el bloque convolucional con los filtros (28, 128, 512) seguido de cinco bloques de identidad con los mismos filtros.

En la quinta fase, añadimos el bloque convolucional con los filtros (256, 256, 1024) seguido de dos bloques de identidad con los mismos filtros.

Lo siguiente, es añadir la capa **AveragePooling2D** con tamaño de pool (2,2). Finalmente, para la capa de salida, se añade una capa **Flatten**, una capa **Dense** con 512 unidades y activación relu y una capa **Dense** con tantas unidades como clases y activación softmax.

5.5. Fusión

Para desarrollar un algoritmo de fusión de clasificaciones de audio y vídeo se han utilizado las técnicas de late fusion y early fusion.

En la fusión tardía (late fusion), mediante serialización se han cogido los resultados de los modelos de audio y vídeo que han dado mejor resultado. Después, se ha implementado una clase que calcula la media ponderada tensores (salidas del modelo), donde el peso puede ser aprendido automáticamente. Se ha añadido la restricción de que los pesos deben sumar 1. Para garantizar esto simplemente tenemos que aplicar un softmax a nuestros pesos. El modelo de fusión es un modelo que recibe dos entradas, por un lado los datos del vídeo y, por otro lado, los datos del audio.

En la fusión temprana (early fusion), se ha creado un modelo base (común a ambas modalidades) y se codificar cada una de las dos entradas en un vector con el modelo base. Finalmente, se concatenan los dos codificadores y se crea el modelo que recibe dos entradas.

Estos modelos no han podido ser probados por falta de tiempo.

5.6. Resultados obtenidos

En esta sección se muestran los resultados que se han obtenido en cada modalidad con las distintas arquitecturas que se han implementado. Todos los resultados son 5-fold cross-validated, es decir, para obtener los resultados que se muestran a continuación, se ha dividido el conjunto de datos en 5 subconjuntos y el modelo se ha entrenado 4 subconjunto, el conjunto restante se utiliza para evaluar el modelo, como se ha explicado en la sección 4.7.

Se ha decidido utilizar 5-fold cross-validation porque esta técnica mejora el modelo mediante la validación de los datos. Mediante este método se garantiza que la puntuación del modelo no está relacionada con la técnica que utilizamos para elegir el conjunto de datos de prueba o de entrenamiento.

Esta sección se divide en distintas subsecciones. En la primera subsección, se presenta la tabla de resultados de reconocimiento facial en la base de datos FER-2013. En la segunda subsección, se muestran los resultados obtenidos en la base de datos RAVDESS. En las siguientes cuatro subsecciones se exponen los resultados obtenidos en las cuatro bases de datos utilizadas para el reconocimiento de emociones en el habla.

5.6.1. Imagen

A continuación se muestran los resultados de los diferentes modelos implementados utilizando la base de datos FER-2013.

Model	Accuracy	F1-Score	Precision	Recall
AlexNet	**84,68**	**84,80**	**85,09**	**84,68**
CNN	80,53	80,48	80,62	80,53
DCNN	84,32	84,21	84,57	84,32
LSTM	59,26	59,21	59,20	59,26
CNN LSTM	68,96	68,83	68,93	68,96
VGG16	81,85	81,77	82,36	81,85
VGG19	73,19	72,13	74,53	73,19
ResNet50	82,39	82,35	82,64	82,39

Tabla 5.2: Resultados de los modelos de imagen en FER-2013 en tanto %.

Como podemos observar en la tabla, AlexNet es el modelo que nos ha dado mejor resultado, alcanzando una exactitud del 84,68 %. En este modelo se ha utilizado el optimizador

SGD[9], learning rate 0,001, loss= sparse categorical crossentropy[10], 30 epoch y batch size=32.

En la siguiente imagen, podemos observar la matriz de confusión obtenida con este modelo.

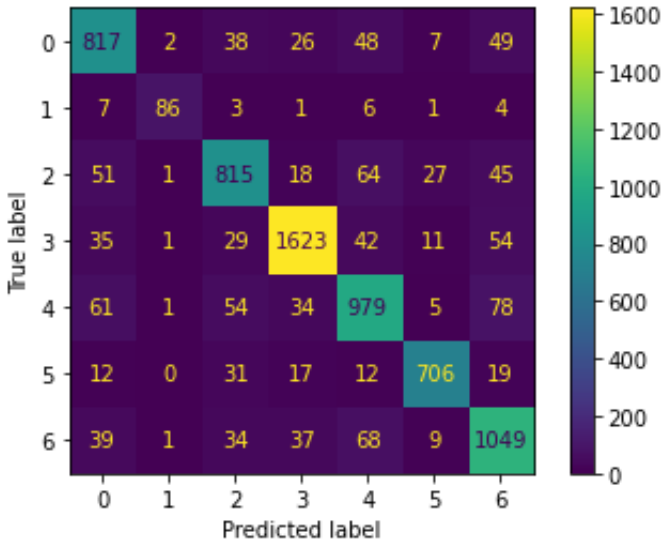


Figura 5.15: Matriz de confusión del modelo AlexNet en FER-2013.

5.6.2. Vídeo

A continuación se muestran los resultados de los diferentes modelos implementados utilizando la base de datos RAVDESS.

Model	Accuracy	F1-Score	Precision	Recall
AlexNet	80,93	81,04	90,32	80,93
CNN	86,74	86,72	87,06	86,74
DCNN	87,43	87,35	88,67	87,43
LSTM	59,26	59,21	59,20	59,26
CNN LSTM	72,15	71,92	73,17	72,15
VGG16	79,04	77,56	81,30	79,04
VGG19	93,65	93,65	94.11	93,65
Inception v3	**97,39**	**97,36**	**97,40**	**97,39**
ResNet50	92,01	91,87	93,12	92,01

Tabla 5.3: Resultados de los modelos de vídeo en RAVDESS en tanto %.

[9]https://keras.io/api/optimizers/sgd/
[10]https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy

Como podemos observar en la tabla, Inception v3 es el modelo que nos ha dado mejor resultado, alcanzando una exactitud del 97,39 %. En este modelo se ha utilizado el optimizador SGD, learning rate 0,001, loss= sparse categorical crossentropy, 30 epoch y batch size=32.

En la siguiente imagen, podemos observar la matriz de confusión obtenida con este modelo.

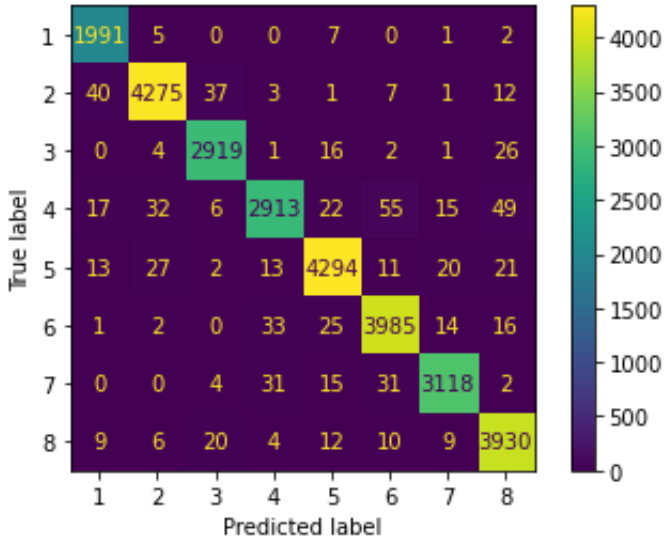


Figura 5.16: Matriz de confusión del modelo Inception v3 en RAVDESS.

5.6.3. Audio

En la siguiente tabla se puede observar los diferentes resultados obtenidos con la base de datos RAVDESS.

Model	Feature	Accuracy	F1-Score	Precision	Recall
AlexNet	Chroma	62,57	61,96	66,28	62,57
	Melspectogram	32,57	29,36	48,07	32,57
	MFCC	58,26	57,64	67,70	58,26
	MFCC Chroma	55,63	51,52	64,85	55,63
	MFCC Melspectrogram	66,67	64,96	72,46	66,67
	MFCC STFT	66,32	64,95	74,56	66,32
	STFT	79,10	78,75	80,08	79,10
CNN	Chroma	78,26	78,27	78,46	78,26
	Melspectogram	66,74	66,42	67,93	66,74
	MFCC	77,57	77,32	78,25	77,57
	MFCC Chroma	78,33	78,40	80,02	78,33
	MFCC Melspectrogram	80,21	80,13	80,82	80,21
	MFCC STFT	**81,25**	**81,25**	**81,96**	**81,25**
	STFT	78,89	78,69	78,95	78,89
CNN LSTM	Chroma	35,28	34,06	36,44	35,28
	Melspectogram	55,90	55,51	56,72	55,90
	MFCC	43,75	41,66	46,70	43,75
	MFCC Chroma	55,97	54,53	61,04	55,97
	MFCC Melspectrogram	64,86	64,86	66,43	64,86
	MFCC STFT	71,74	71,24	72,95	71,74
	STFT	69,24	69,36	67,93	66,74
LSTM	Chroma	48,40	48,00	48,89	48,40
	Melspectogram	65,49	65,52	68,14	65,49
	MFCC	73,54	73,49	75,06	73,54
	MFCC Chroma	68,54	68,68	71,80	68,54
	MFCC Melspectrogram	66,04	65,67	68,10	66,04
	MFCC STFT	74,79	74,69	77,68	74,79
	STFT	71,25	71,39	72,34	71,25
MLP	Chroma	18,75	18,49	18,68	18,75
	Melspectogram	30,35	29,21	30,00	30,35
	MFCC	47,29	46,73	48,75	47,29
	MFCC Chroma	44,17	43,34	46,00	44,17
	MFCC Melspectrogram	43,19	42,57	45,51	43,19
	MFCC STFT	38,40	37,36	39,93	38,40
	STFT	17,57	15,93	18,25	17,57

Tabla 5.4: Resultados en tanto % sobre base de datos RAVDESS.

Como podemos observar en la tabla, el modelo CNN con las técnicas MFCC y STFT nos ha dado mejor resultado, alcanzando una exactitud del 81,25 %. En este modelo se ha utilizado el optimizador Adam, learning rate 0,001, loss= sparse categorical crossentropy, 30 epoch y batch size=32.

En la siguiente imagen se muestra la matriz de confusión del modelo.

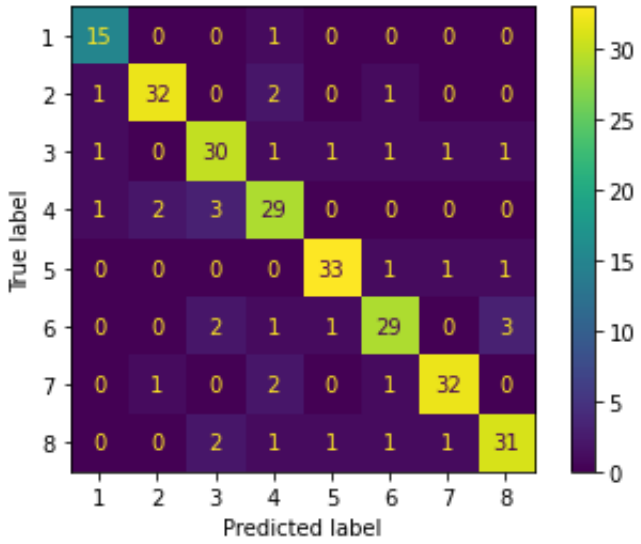


Figura 5.17: Matriz de confusión del modelo CNN MFCC STFT en RAVDESS.

En la siguiente tabla se puede observar los diferentes resultados obtenidos con la base de datos TESS.

Model	Feature	Accuracy	F1-Score	Precision	Recall
AlexNet	Chroma	62,57	61,96	66,28	66,28
	Melspectogram	32,57	29,36	,48,07	32,57
	MFCC	58,26	57,64	67,70	58,26
	MFCC Chroma	55,63	51,52	64,85	55,63
	MFCC Melspectrogram	66,67	64,96	72,46	66,67
	MFCC STFT	66,32	64,95	74,56	66,32
	STFT	79,10	78,75	80,08	79,10
CNN	Chroma	96,68	96,67	96,77	96,68
	Melspectogram	98,04	98,04	98,16	98,04
	MFCC	98,71	98,71	98,75	98,71
	MFCC Chroma	98,71	98,71	98,75	98,71
	MFCC Melspectrogram	99,25	99,25	99,26	99,25
	MFCC STFT	98,93	98,93	98,96	98,93
	STFT	95,21	95,18	95,27	95,21
CNN LSTM	Chroma	93,57	93,57	94,01	93,57
	Melspectogram	99,18	99,18	99,20	99,18
	MFCC	98,61	98,58	98,76	98,61
	MFCC Chroma	98,43	98,41	98,55	98,43
	MFCC Melspectrogram	99,61	99,61	99,61	99,61
	MFCC STFT	98,39	98,38	98,64	98,39
	STFT	96,00	96,02	96,25	96,00
LSTM	Chroma	95,18	94,53	94,78	94,57
	Melspectogram	99,32	99,32	99,35	99,32
	MFCC	99,50	99,50	99,51	99,50
	MFCC Chroma	99,39	99,39	99,40	99,39
	MFCC Melspectrogram	**99,68**	**99,68**	**99,68**	**99,68**
	MFCC STFT	99,32	99,32	99,32	99,32
	STFT	95,36	95,18	95,70	95,36
MLP	Chroma	83,54	83,51	83,90	83,54
	Melspectogram	94,79	94,78	95,03	94,79
	MFCC	94,46	99,46	94,56	94,46
	MFCC Chroma	94,75	94,75	94,82	94,75
	MFCC Melspectrogram	97,71	97,72	97,78	97,71
	MFCC STFT	86,43	86,11	86,97	86,43
	STFT	32,86	31,32	32,27	32,86

Tabla 5.5: Resultados de los modelos de audio en tanto % sobre base de datos TESS.

Como podemos observar en la tabla, el modelo LSTM con las técnicas MFCC y Melspectogram nos ha dado mejor resultado, alcanzando una exactitud del 99,68 %. En este

modelo se ha utilizado el optimizador Adam, learning rate 0,001, loss= sparse categorical crossentropy, 30 epoch y batch size=32.

En la siguiente imagen se muestra la matriz de confusión del modelo.

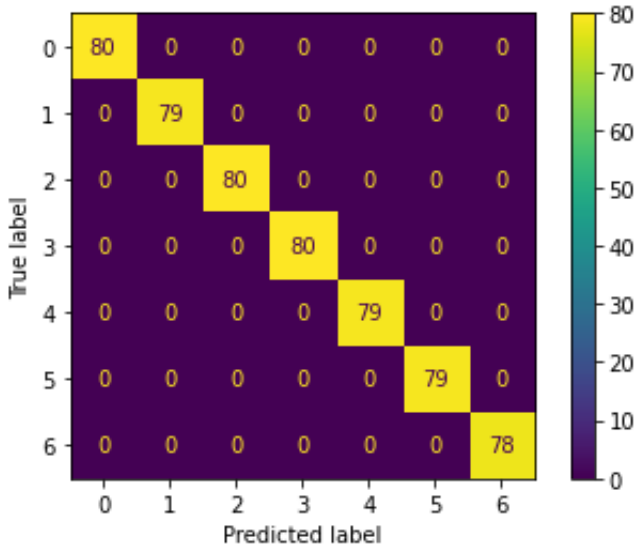


Figura 5.18: Matriz de confusión del modelo CNN MFCC Melspectogram en TESS.

En la siguiente tabla se puede observar los diferentes resultados obtenidos con la base de datos SAVEE.

Model	Feature	Accuracy	F1-Score	Precision	Recall
AlexNet	Chroma	68,00	66,01	67,02	68,00
	Melspectogram	66,25	73,00	63,16	66,25
	MFCC	73,96	73,56	79,44	73,96
	MFCC Chroma	77,08	75,70	82,56	77,08
	MFCC Melspectrogram	79,38	79,54	82,62	79,375
	MFCC STFT	74,37	73,24	76,97	74,38
	STFT	79,58	78,42	79,94	79,58
CNN	Chroma	83,75	83,46	83,43	83,75
	Melspectogram	66,67	66,30	69,25	66,67
	MFCC	**87,29**	**87,05**	**87,47**	**87,29**
	MFCC Chroma	85,21	84,85	84,99	85,21
	MFCC Melspectrogram	71,04	71,09	73,04	71,04
	MFCC STFT	80,83	79,71	81,41	80,83
	STFT	73,54	72,76	73,95	73,54
CNN LSTM	Chroma	43,75	40,34	42,37	43,75
	Melspectogram	63,33	60,96	63,46	63,33
	MFCC	60,00	57,40	62,39	60,00
	MFCC Chroma	71,46	69,41	71,05	71,46
	MFCC Melspectrogram	70,83	70,24	74,35	70,83
	MFCC STFT	83,13	82,37	84,21	83,13
	STFT	81,88	81,01	82,59	81,88
LSTM	Chroma	51,67	49,46	51,22	51,67
	Melspectogram	85,42	85,15	85,59	85,42
	MFCC	75,83	75,58	77,05	75,83
	MFCC Chroma	73,75	73,21	76,18	73,75
	MFCC Melspectrogram	77,08	76,71	79,37	77,08
	MFCC STFT	84,79	84,70	88,14	84,79
	STFT	82,08	81,54	82,23	82,08
MLP	Chroma	25,83	23,61	23,26	25,83
	Melspectogram	29,38	28,34	30,73	29,38
	MFCC	36,88	35,43	39,29	36,88
	MFCC Chroma	33,96	33,13	34,49	33,96
	MFCC Melspectrogram	35,42	32,66	34,71	35,42
	MFCC STFT	32,29	28,60	29,81	32,29
	STFT	28,13	22,79	25,76	28,13

Tabla 5.6: Resultados de los modelos de audio en tanto % sobre base de datos SAVEE.

Como podemos observar en la tabla, el modelo CNN con la técnica MFCC nos ha dado mejor resultado, alcanzando una exactitud del 87,29 %. En este modelo se ha utilizado el

optimizador Adam, learning rate 0,001, loss= sparse categorical crossentropy, 30 epoch y batch size=32.

En la siguiente imagen podemos observar la matriz de confusión del modelo.

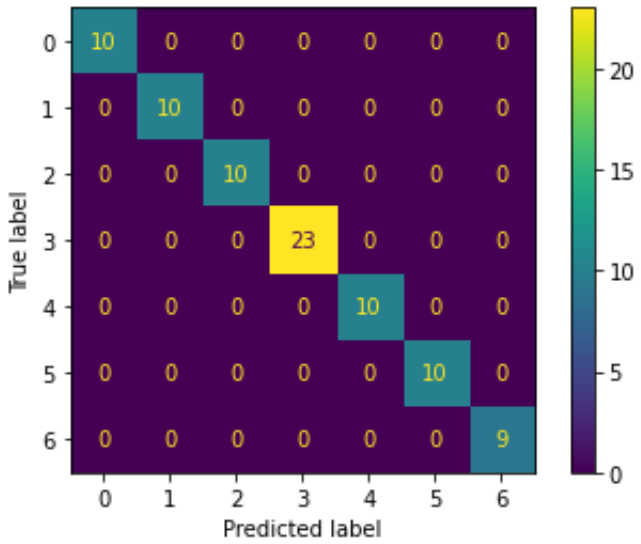


Figura 5.19: Matriz de confusión del modelo CNN MFCC en SAVEE.

En la siguiente tabla se puede observar los diferentes resultados obtenidos con la base de datos Crema-D.

Model	Feature	Accuracy	F1-Score	Precision	Recall
AlexNet	Chroma	62,88	62,75	63,21	62,88
	Melspectogram	49,33	46,28	63,66	49,33
	MFCC	67,34	67,24	68,62	67,34
	MFCC Chroma	65,00	64,60	67,09	65,00
	MFCC Melspectrogram	63,99	63,62	69,75	63,99
	MFCC STFT	65,63	64,12	71,57	65,63
	STFT	**77,03**	**76,80**	**77,86**	**77,03**
CNN	Chroma	73,99	73,85	74,09	73,99
	Melspectogram	59,69	59,44	59,63	59,69
	MFCC	71,67	71,50	72,15	71,67
	MFCC Chroma	74,85	74,73	75,24	74,85
	MFCC Melspectrogram	63,35	63,26	63,86	63,35
	MFCC STFT	69,60	69,47	70,44	69,60
	STFT	69,42	69,39	69,58	69,42
CNN LSTM	Chroma	35,13	34,51	35,41	35,13
	Melspectogram	54,52	53,96	55,29	54,52
	MFCC	55,54	54,88	57,90	55,54
	MFCC Chroma	57,00	56,43	59,21	57,00
	MFCC Melspectrogram	58,96	58,32	59,80	58,96
	MFCC STFT	63,96	63,81	65,07	63,96
	STFT	59,89	59,94	62,70	59,89
LSTM	Chroma	42,29	41,95	42,45	42,29
	Melspectogram	69,53	69,38	70,02	69,53
	MFCC	65,62	65,66	67,32	65,62
	MFCC Chroma	65,51	65,36	66,59	65,51
	MFCC Melspectrogram	66,91	66,83	68,39	66,91
	MFCC STFT	68,55	68,46	69,46	68,55
	STFT	72,03	71,82	71,98	72,03
MLP	Chroma	22,86	22,78	22,80	22,86
	Melspectogram	33,85	32,73	32,70	33,85
	MFCC	41,71	40,50	43,09	41,71
	MFCC Chroma	40,23	38,56	42,54	40,23
	MFCC Melspectrogram	41,60	40,63	41,54	41,60
	MFCC STFT	34,90	34,90	35,62	34,90
	STFT	20,44	19,24	21,29	20,44

Tabla 5.7: Resultados de los modelos de audio en tanto% sobre base de datos Crema-D.

Como podemos observar en la tabla, el modelo AlexNet con la técnica STFT nos ha dado mejor resultado, alcanzando una exactitud del 77,03 %. En este modelo se ha utilizado el

optimizador Adam, learning rate 0,001, loss= sparse categorical crossentropy, 30 epoch y batch size=32.

En la siguiente imagen podemos observar la matriz de confusión del modelo.

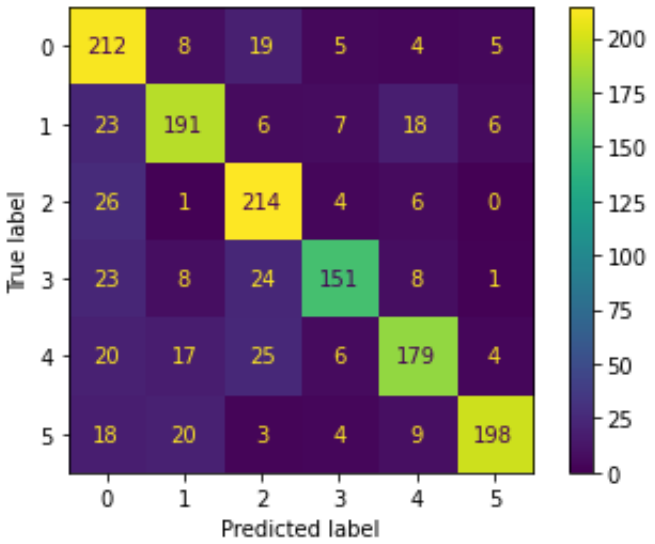


Figura 5.20: Matriz de confusión del modelo AlexNet STFT en Crema-D.

5.6.4. Comparación de resultados

En esta sección se hace una comparativa entre los resultados obtenidos en este proyecto y los mejores resultados que se han registrado hasta el momento, para ver si se ha conseguido mejorarlos o no.

Modalidad	Base de datos	Modelo Mejor resultado registrado	Modelo Mejor resultado proyecto
Imagen	FER-2013	ResMaskingNet con 6 CNNs [21] 76,82 %	AlexNet 84,68 %
Vídeo	RAVDESS	Intermediate-Fusion Transformer [22] 86,70 %	Inception v3 97,39 %
Audio	RAVDESS	TIM-Net [27] 92,08 %	CNN MFCC STFT 81,25 %
	TESS	MLP classifier [29] 97,2 %	LSTM MFCC Melspectogram 99,68 %
	SAVEE	TIM-Net [27] 87,71 %	CNN MFCC 87,29 %
	Crema-D	LeRaC [28] 70,95 %	AlexNet STFT 77,07 %

Tabla 5.8: Comparación entre los mejores resultados registrados por otros autores y los resultados del proyecto.

Como podemos observar en la tabla, el la mayoría de los casos se han conseguido mejorar los resultados de otros trabajos previos. Se han obtenido peores resultados en la modalidad de audio con las bases de datos RAVDESS y SAVEE.

6. CAPÍTULO

Desviaciones del proyecto

6.1. Dedicación real

En la siguiente tabla se muestra la dedicación real en horas para cada bloque de tareas:

Bloque	Bloque específico	Dedicación	Dedicación bloque
Investigación	Búsqueda de información	45h	80h
	Estado del arte	35h	
Implementación	Desarrollo de prototipos	85h	135h
	Experimentación	50h	
Gestión	Planificación	10h	45h
	Seguimiento y control	35h	
Memoria	Bibliografía	50h	110h
	Resultados	60h	
Total			370h

Tabla 6.1: Dedicación real en horas por cada bloque de tareas.

6.2. Desviación entre la dedicación estimada y la dedicación real

En la siguiente tabla se muestra la desviación en horas entre la dedicación que se había estimado y la dedicación real.

Bloque	Dedicación estimada	Dedicación real	Desviación
Investigación	70h	80h	+10h
Implementación	105h	135h	+30h
Gestión	40h	45h	+5h
Memoria	100h	110h	+10h
Total			+55h

Tabla 6.2: Desviación en horas por cada bloque de tareas.

Como se puede observar, al final hemos necesitado 370 horas para poder completar el proyecto, 55 horas más de las previstas. Esto se debe principalmente a la implementación y las pruebas de los algoritmos. Como estos apartados son realmente importantes en el proyecto, no nos ha importado incrementar su tiempo de dedicación aún sabiendo que de esta forma se podría no llegar a cumplir la planificación inicial. En cuanto a los bloques de investigación y memoria, se han invertido 10 horas más de las previstas en cada bloque. La investigación ha sido, en parte, una de las bases de este proyecto ya que ha sido un campo desconocido y era importante profundizar en todos los conceptos para su correcta comprensión y poder realizar el proyecto.

7. CAPÍTULO

Conclusiones

En este TFG, se ha conseguido implementar distintas redes neuronales para el reconocimiento de emociones en distintas modalidades. Se han probado las redes implementadas con distintos parámetros y distintas características y, aunque en algunos casos los resultados obtenidos no han sido muy buenos, en otros se ha podido observar un buen rendimiento.

La base de este proyecto es que se han implementado distintas arquitecturas de redes neuronales y se han hecho numerosas pruebas para ver en cada modalidad cuál es la que nos puede dar mejores resultados. En este proyecto, se ha conseguido mejorar los resultados con respecto a trabajos previos (como se ha podido observar en 5.6.4). Por ejemplo, en el reconocimiento de emociones en imágenes sobre la base de datos FER-2013 casi se ha alcanzado un 8 % más de exactitud (7,86 %, para ser exactos). En la modalidad de vídeo sobre la base de datos RAVDESS se ha conseguido mejorar el resultado en un 10,69 %. Y, finalmente, en la modalidad de audio se ha conseguido mejorar los resultados obtenidos en las bases de datos TESS y Crema-D (2,48 % y 6,08 % respectivamente).

Ha sido una grata oportunidad poder realizar este proyecto. Por una parte me ha resultado un campo muy interesante. Por otra parte, he podido trabajar con redes neuronales que, están en auge hoy en día y tienen muchas aplicaciones.

Como trabajo futuro, a parte de probar los modelos de fusión implementados, para poder mejorar los modelos se puede utilizar aumento de datos. Por ejemplo, rotar y distorsionar imágenes, modificar el tono del audio, modificar la duración del audio, etc.

Bibliografía

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. (2015). Deep Residual Learning for Image Recognition.

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. (2012). ImageNet Classification with Deep Convolutional Neural Networks.

[3] F. Eyben and e. a. M. Wöllmer. (2010). Emotion on the road—necessity, acceptance, and feasibility of affective) computing in the car. Advances in Human-Computer Interaction.

[4] E. Cambria. (2016). Affective computing and sentiment analysis. IEEE Intelligent Systems.

[5] Livingstone SR, Russo FA. (2018). The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391. https://doi.org/10.1371/journal.pone.0196391.

[6] Pichora-Fuller, M. Kathleen; Dupuis, Kate. (2020). Toronto emotional speech set (TESS). https://doi.org/10.5683/SP2/E8H2MF, Borealis, V1.

[7] Shah, Ayush & Kattel, Manasi & Nepal, Araju & Shrestha, D.. (2019). Chroma Feature Extraction.

[8] Lahat et al. (2015). Multimodal Data Fusion : An Overview of Methods , Challenges and Prospects.Proceedings of the IEEE.

[9] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi. (2013). Multisensor data fusion: A review of the state-of-the-art. Inf. Fusion.

[10] H. P. Martínez and G. N. Yannakakis. (2014). Deep Multimodal Fusion.

[11] R. Dhanesh and T. Graham W. (2017). Deep Multimodal Learning: A Survey on Recent Advances and Trends. IEEE Signal Process.

[12] B. Vinay and B. S. Shreyas. (2006). Face Recognition Using Gabor Wavelets.

[13] C. Shan, S. Gong and P. Mcowan. (2005). Robust facial expression recognition using local binary patterns.

[14] A. Balakrishnan and A. Rege. (2017). Reading Emotions from Speech using Deep Neural Networks.

[15] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B.Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee et al. (2013). Challenges in representation learning: A report on three machine learning contests. International Conference on Neural Information Processing. Springer.

[16] Y. Tang. (2013). Deep Learning using Support Vector Machines. International Conference on Machine Learning (ICML) Workshops.

[17] S. Li and W. Deng. (2018). Deep facial expression recognition: A survey.

[18]] Pramerdorfer, C., Kampel, M. (2016). Facial expression recognition using convolutional neural networks: state of the art. Preprint arXiv:1612.02903v1.

[19] Z. Zhang, P. Luo, C.-C. Loy, and X. Tang. (2015). Learning Social Relation Traits from Face Images. Proc. IEEE Int. Conference on Computer Vision (ICCV).

[20] Foundations of human computing: facial expression and emotion, Cohn, Jeffrey F. (2006). Proceedings of the 8th international conference on Multimodal interfaces.

[21] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. (2013). Challenges in Representation Learning: A report on three machine learning contests. arXiv:1307.0414v1.

[22] Kateryna Chumachenko, Alexandros Iosifidis and Moncef Gabbouj. (2022). Self-attention fusion for audiovisual emotion recognition with incomplete data. arXiv:2201.11095v1.

[23] J. Rong, G. Li, and Y.-P. P. Chen. (2009). Acoustic feature selection for automatic emotion recognition from speech. Information Processing & Management.

[24] G. E. Hinton and R. R. Salakhutdinov. (2006). Reducing the dimensionality of data with neural networks. Science, vol. 313, no. 5786.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. (2012). Imagenet classification with deep convolutional neural networks, in Proceedings of the Advances in Neural Information Processing.

[26] A. U A and K. V K. Speech Emotion Recognition-A Deep Learning Approach. (2021). Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud). I-SMAC.

[27] Jiaxin Ye, Xincheng Wen, Yujie Wei, Yong Xu, Kunhong Liu, Hongming Shan. (2022).TEMPORAL MODELING MATTERS: A NOVEL TEMPORAL EMOTIONAL MODELING APPROACH FOR SPEECH EMOTION RECOGNITION. arXiv:2211.08233v1.

[28] Florinel-Alin Croitoru, Nicolae-Cat Ristea, Radu Tudor Ionescu, Nicu Sebe. (2022). LeRaC: Learning Rate Curriculum. arXiv:2205.09180v2.

[29] Akinpelu, S., Viriri, S. (2022). Robust Feature Selection-Based Speech Emotion Classification Using Deep Transfer Learning. Appl. Sci.