
Reconocimiento de emociones faciales utilizando Deep Learning

Facial Emotion Recognition using Deep Learning



**Trabajo de Fin de Grado
Curso 2023–2024**

Autor

Francisco Calvo González

Luis Morales Júlvez

Jaime Costas Insua

Director

Mercedes García Merayo

Manuel Méndez Hurtado

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

Reconocimiento de emociones faciales utilizando Deep Learning

Facial Emotion Recognition using Deep Learning

Trabajo de Fin de Grado en Ingeniería Informática

Autor

Francisco Calvo González
Luis Morales Júlvez
Jaime Costas Insua

Director

Mercedes García Merayo
Manuel Méndez Hurtado

Convocatoria: *Junio 2024*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

27 de mayo de 2024

Dedicatoria

*Al grupo LDC, por formar parte de todos los
recuerdos de estos maravillosos años.*

Agradecimientos

Queremos darle las gracias a Paloma, por darnos cobijo, alimento y cariño en unos de los días mas importantes y difíciles de nuestras carreras; a José, por darnos las mejores comidas que hemos tenido en mucho tiempo; y a cada uno de nosotros, por aguantarnos cuando nadie nos aguantaba, por animarnos cuando ni nosotros mismos confiábamos en nosotros y por ser los mejores compañeros de trabajo que se podría tener.

Resumen

Reconocimiento de emociones faciales utilizando Deep Learning

El reconocimiento de expresiones faciales es una tarea crucial en el campo de la visión artificial, fundamental en la interacción persona-ordenador. En los últimos años, se ha demostrado la eficiencia de las redes neuronales para llevar a cabo esta tarea.

En este trabajo, presentamos una red neuronal convolucional para el reconocimiento de expresiones faciales en fotografías de seres humanos en situaciones cotidianas. Asimismo, profundizamos en diferentes técnicas para mejorar el aprendizaje de nuestro modelo, empleando nuevos algoritmos no aplicados en este problema hasta ahora.

Para ello, comenzamos introduciendo varias iteraciones de una arquitectura de red neuronal convolucional, con ligeras variaciones entre sí. A continuación, entrenamos cada una de ellas aplicando técnicas de preprocesamiento y de aumento de datos para conseguir modelos con una alta capacidad clasificativa. Finalmente, demostramos la precisión general de nuestro modelo y lo comparamos con los modelos más efectivos en la actualidad.

Palabras clave

Redes neuronales, convolución, VGG, AffectNet, LeakyReLU, Schedule-free Adam, aumento de datos, costes ponderados.

Abstract

Facial Emotion Recognition using Deep Learning

Facial expression recognition is a crucial task in the field of computer vision, fundamental for human-computer interaction. Recent years have witnessed and increased efficiency of neuronal networks to perform this task.

Here, we present a deep convolutional neural network for facial expression recognition in photographs of humans in everyday situations. We also delve into different techniques to improve the learning of our model, making use of new algorithms not applied in this problem so far.

To do so, we start by introducing several iterations of a convolutional neural network architecture, with slight variations. We then train each of them by applying preprocessing and data augmentation techniques to achieve models with high classification capability. We demonstrate the overall accuracy of our model and compare it with current most effective models.

Keywords

Neural Networks, convolution, VGG, Affectnet, LeakyReLU, Schedule-free Adam, data augmentation, weighted costs.

Índice

1. Introducción	1
1.1. Objetivo	1
1.2. Planificación	1
1. Introduction	3
1.1. Objective	3
1.2. Experimental plan	3
2. Introducción a las redes neuronales	5
2.1. Red neuronal	5
2.2. Entrenamiento	7
2.3. Redes convolucionales	8
3. Estado del Arte	13
3.1. Metodologías clásicas	13
3.2. Redes neuronales convolucionales	16
3.3. Arquitecturas mixtas	18
4. Nuestra propuesta	21
4.1. VGG10a	22
4.2. VGG10b	22
4.3. VGG12	23
4.4. VGG14	24
4.5. VGG17	25
5. Experimentación	27
5.1. Dataset	27
5.2. Preprocesamiento	29
5.3. Optimización	30
5.3.1. Función de coste	30
5.3.2. Optimizadores de gradiente	31
5.4. Estrategias de aprendizaje	32
5.4.1. Coste ponderado	32
5.4.2. Aumento de datos	33
5.4.3. Dropout	34

6. Métricas de evaluación	35
6.1. Exactitud	35
6.2. Precisión	35
6.3. Recall	36
6.4. Matriz de confusión	36
6.5. Métodos de evaluación	37
7. Resultados	39
Conclusiones y Trabajo Futuro	45
Conclusions and Future Work	47
Contribuciones Personales	49
Bibliografía	53

Índice de figuras

2.1.	Representación de una red neuronal.	5
2.2.	Funciones lineales comunes.	6
2.3.	Estructura de un perceptrón básico.	7
2.4.	Ilustración del descenso de gradiente estocástico.	8
2.5.	Efecto del <i>overshooting</i> en el SGD.	8
2.6.	Arquitectura típica de una CNN (Koushik, 2023).	9
2.7.	Aplicación de un filtro convolucional sobre una entrada (Akhtar et al., 2022).	9
2.8.	Mapas de características de una CNN para reconocer perros (vaishnav, 2020).	10
2.9.	Ejemplo de Min Pooling (Soto, 2022).	11
2.10.	Ejemplo de Max Pooling (Soto, 2022).	11
2.11.	Ejemplo de Average Pooling (Soto, 2022).	11
3.1.	Aplicación de un modelo físico para extraer unidades de acción: imagen original, modelo geométrico, y extracción (Vadapalli, 2011).	14
3.2.	Arquitectura AlexNet (Nabil, 2023).	16
3.3.	Arquitectura VGGNet-16.	17
3.4.	Arquitectura GoogLeNet (Nabil, 2023).	17
3.5.	Arquitectura ResNet (Nabil, 2023).	18
3.6.	Ejemplo de CNN y LSTM (Akhtar et al., 2022).	19
3.7.	Estructura de la arquitectura DDAMFN (Zhang et al., 2023).	19
3.8.	Extracción de características y puntos clave de una imagen (ju Mao et al., 2023).	20
3.9.	Estructura del modelo BoVW (Georgescu et al., 2019).	20
4.1.	Arquitectura VGGNet-16.	21
4.2.	Arquitectura VGG10a.	22
4.3.	Arquitectura VGG10b.	23
4.4.	Arquitectura VGG12.	24
4.5.	Arquitectura VGG14.	24
4.6.	Arquitectura VGG17.	25
5.1.	Transformación a escala de grises.	29
5.2.	Ejemplo de tamaños: 225x225, 128x128 y 64x64.	30

5.3.	Evolución de la función de coste con diferentes optimizadores en una red neuronal multicapa MNIST (Sepúlveda, 2023)	31
5.4.	Efecto del sobreaprendizaje en un modelo.	32
5.5.	Distribución de clases en AffectNet.	33
5.6.	Imágenes antes y después del preprocesamiento y aumento de datos. .	34
5.7.	Error de varias redes con y sin <i>dropout</i> (Srivastava et al., 2014a)	34
6.1.	Ejemplo de matriz de confusión.	36
7.1.	Evolución de la exactitud durante el entrenamiento de VGG10a.	40
7.2.	Evolución de la función de coste durante el entrenamiento de VGG10a. .	40
7.3.	Evolución de la exactitud durante el entrenamiento de VGG10b.	40
7.4.	Evolución de la función de coste durante el entrenamiento de VGG10b. .	40
7.5.	Evolución de la exactitud durante el entrenamiento de VGG12.	41
7.6.	Evolución de la función de coste durante el entrenamiento de VGG12. .	41
7.7.	Evolución de la exactitud durante el entrenamiento de VGG14.	41
7.8.	Evolución de la función de coste durante el entrenamiento de VGG14. .	41
7.9.	Evolución de la exactitud durante el entrenamiento de VGG17.	42
7.10.	Evolución de la función de coste durante el entrenamiento de VGG17. .	42
7.11.	Matriz de confusión de VGG10a con el subconjunto de validación público.	42
7.12.	Matriz de confusión de VGG10b con el subconjunto de validación público.	42
7.13.	Matriz de confusión de VGG12 con el subconjunto de validación público. .	43
7.14.	Matriz de confusión de VGG14 con el subconjunto de validación público. .	43
7.15.	Matriz de confusión de VGG17 con el subconjunto de validación público. .	43
7.16.	Confusión entre imágenes de las clases Rabia y Asco.	43
7.17.	Imágenes de la clase Neutral que pueden interpretarse como otras clases.	44

Índice de tablas

3.1.	Unidades de acción principales.	15
5.1.	Distribución de datos de AffectNet.	28
5.2.	Ejemplos de distintas clases de AffectNet.	28
5.3.	Asignación de coeficientes de coste para cada clase.	33
7.1.	Exactitud de cada modelo por clases con el conjunto de prueba. . . .	39

Capítulo 1

Introducción

Desde el comienzo de la informática moderna, la creación de sistemas de inteligencia artificial ha sido un tema de gran interés. Los avances en la capacidad de computación, el desarrollo de hardware dedicado al procesamiento de redes neuronales en dispositivos populares, y el reciente interés de la sociedad en el campo abren muchas oportunidades para aplicar redes neuronales a todo tipo de problemas.

Una emoción es un estado afectivo que surge como respuesta a estímulos internos o externos. En los seres humanos, uno de los principales efectos de las emociones es un cambio en la expresión facial. Estos cambios son en gran parte universales (Ekman y Friesen, 1971) y, por ello, la capacidad de reconocer expresiones faciales, comúnmente denominada *Facial Expression Recognition* (FER), se considera una de las habilidades sociales más importantes. Esta cualidad, innata para nuestra especie, comienza a aparecer en bebés con apenas cuatro meses de vida (Nelson, 1987).

La automatización de esta tarea tiene una gran cantidad de aplicaciones en nuestra sociedad, incluyendo marketing, atención al cliente, interacción humano-ordenador, detección de trastornos mentales, por citar únicamente algunas.

1.1. Objetivo

El objetivo de este trabajo es la proposición de una arquitectura de una red neuronal convolucional profunda capaz de reconocer expresiones faciales en fotografías de individuos en el mundo real. Esta arquitectura, diseñada para competir con los modelos de mayor precisión de la actualidad, debe conseguir un rendimiento similar al de un ser humano. Además, trataremos de reducir la complejidad del modelo en la medida de lo posible, con el propósito de poder utilizarlo en dispositivos populares.

1.2. Planificación

Para llevar a cabo este proyecto, utilizamos un enfoque descendente o *top-down*. Dividimos su desarrollo en cuatro fases claramente diferenciadas, en las que profundizamos en el proyecto de manera gradual:

1. **Investigación.** Inicialmente, analizamos la tarea de reconocimiento de ex-

presiones faciales y las diferentes estrategias utilizadas para llevarla a cabo hasta la fecha. Además, contemplamos todas las herramientas a nuestro alcance para llevar a cabo el proyecto, así como anticipamos posibles limitaciones y problemas potenciales.

2. **Propuesta de trabajo.** Una vez establecida la tarea a completar y sus implicaciones y desafíos, fijamos el alcance del proyecto, los subobjetivos a cumplir, y la estrategia que seguiremos. Diseñamos también varias arquitecturas con las que realizaremos varios experimentos en la siguiente fase.
3. **Experimentación.** Estando ya marcada la ruta a seguir, comenzamos intentando replicar los resultados obtenidos por otros investigadores. Tras solventar varios problemas inesperados relacionados con las herramientas y el entorno de experimentación, implementamos las arquitecturas propuestas en la etapa anterior. Una vez entrenadas, recolectamos una serie de métricas para estudiar la precisión del modelo.
4. **Análisis de resultados.** Utilizando las métricas obtenidas en el apartado anterior, estudiamos el rendimiento de las diferentes arquitecturas propuestas, y seleccionamos una de ellas. Finalmente, comparamos la arquitectura seleccionada con los modelos de vanguardia, y evaluamos el grado de cumplimiento de nuestros subobjetivos propuestos.

Chapter 1

Introduction

Since the beginning of modern computing, the creation of artificial intelligence systems has been a topic of great interest. Advances in computing power, the development of hardware dedicated to neural network processing in popular devices, and the recent societal interest in the field open up many opportunities to apply neural networks to all kinds of problems.

An emotion is an affective state that arises in response to internal or external stimuli. In humans, one of the main effects of emotions is a change in facial expression. These changes are largely universal (Ekman y Friesen, 1971), and thus the ability to recognize these expressions, commonly referred to as Facial Expression Recognition (FER), is considered one of the most important social skills. This quality, innate to our species, begins to appear in infants as young as four months old (Nelson, 1987).

The automation of this task has a myriad of applications in our society, including marketing, customer service, human-computer interaction, detection of mental disorders, to name but a few.

1.1. Objective

The objective of this work is to propose a deep convolutional neural network architecture capable of recognizing facial expressions in photographs of real-world individuals. This architecture, designed to compete with today's most accurate models, must achieve human-like performance. Moreover, we will attempt to reduce the complexity of the model as much as possible, with the purpose of being able to use it in popular devices.

1.2. Experimental plan

To carry out this project, we use a top-down approach. We divide its development into four clearly differentiated phases, in which we gradually develop the project:

1. **Research.** We start by analyzing the task of facial expression recognition and

the different strategies used to conduct it to date. In addition, we contemplate all the tools at our disposal to perform the project, as well as anticipate the possible limitations and potential problems.

2. **Work proposal.** Once we understand the task to be completed and its implications and challenges, we establish the scope of the project, the subobjectives to be met, and the strategy to follow. We also design several architectures with which we will perform several experiments in the next phase.
3. **Experimentation.** Having already marked the experimental plan, we begin by attempting to replicate the results obtained by other researchers. After solving several unexpected problems related to the tools and the experimentation environment, we implement the architectures proposed in the previous stage. Once trained, we collect a series of metrics to study the accuracy of the model.
4. **Results Analysis.** Using the metrics collected in the previous section, we study the performance of the different proposed architectures, and select one of them. Finally, we compare the selected architecture with the state-of-the-art models, and evaluate the degree of achievement with our subobjectives.

Capítulo 2

Introducción a las redes neuronales

Para poder describir en detalle los aspectos más técnicos de este trabajo, es necesario tener una compresión básica sobre el funcionamiento de las redes neuronales. Este capítulo proporciona una breve historia sobre las redes neuronales, y aborda estos conceptos esenciales.

2.1. Red neuronal

Una red neuronal es un método de machine learning de aprendizaje profundo (Figura 2.1). Estas redes, introducidas por primera vez en los años cuarenta, son modelos simplificados del cerebro humano capaces de extraer y generalizar conocimiento (McCulloch y Pitts, 1943).

Las redes neuronales están formadas por una serie de nodos o neuronas artificiales que imitan las neuronas de un cerebro humano. Estas neuronas se conectan entre sí, imitando la sinapsis de un cerebro. Cada neurona recibe señales de las neuronas conectadas a ella, las procesa y envía una señal al resto de neuronas a las que está conectada. La señal es un número real, y la salida se calcula mediante una función no lineal de la suma de sus entradas denominada función de activación. La fuerza o intensidad de cada conexión entre dos neuronas viene determinada por un peso, un valor que se ajusta durante el entrenamiento.

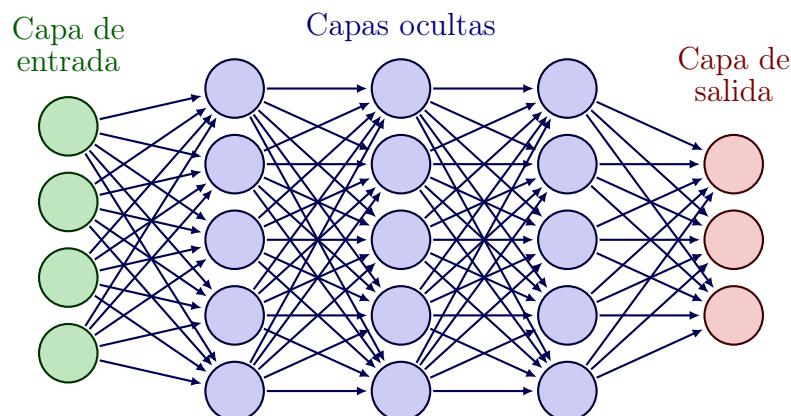


Figura 2.1: Representación de una red neuronal.

La función de activación se encarga de calcular la salida de la neurona. Existen múltiples funciones de activación diferentes, siendo la unidad lineal rectificada o ReLU la más común de todas ellas (Figura 2.2).

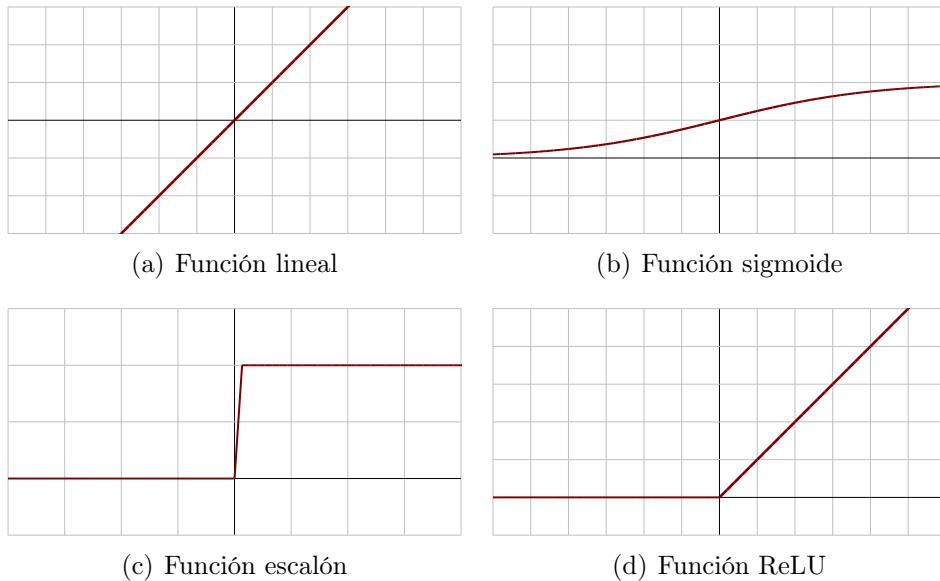


Figura 2.2: Funciones lineales comunes.

Las neuronas suelen clasificarse en capas. Para procesar una entrada, la señal viaja desde la primera capa (capa de entrada) hasta la última (capa de salida), pasando por las capas intermedias (capas ocultas). Una red con dos o más capas ocultas suele clasificarse como una red neuronal profunda.

El perceptrón de Rosenblatt (1958) es una de las primeras implementaciones de redes neuronales capaz de clasificar entradas en dos categorías. La estructura del perceptrón, representada en la Figura 2.3, está compuesta por una capa de entrada con una o más neuronas y una función de activación que determina el resultado. Pese a parecer una investigación prometedora, el interés decrece considerablemente cuando Minsky y Papert (1969) descubren que el perceptrón no puede procesar circuitos simples como el XOR.

Tras un periodo de pocos avances, comúnmente denominado como el segundo invierno de la inteligencia artificial, las innumerables aplicaciones de las redes neuronales comienzan a descubrirse al comienzo del siglo XXI, cuando la capacidad de computación es por primera vez suficiente para poder soportar grandes redes neuronales.

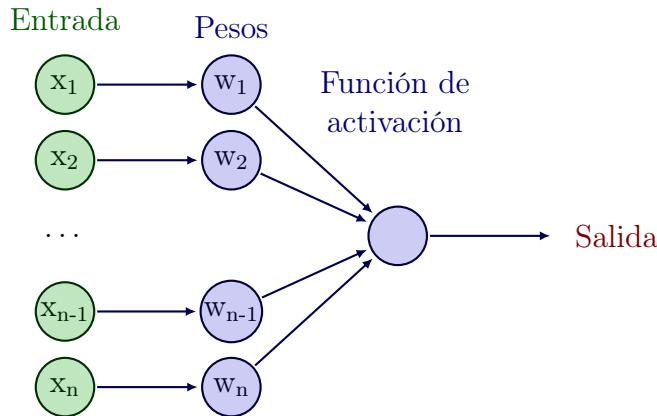


Figura 2.3: Estructura de un perceptrón básico.

2.2. Entrenamiento

Los pesos de cada una de las conexiones de una red neuronal son los parámetros que dictan su comportamiento; es decir, las permutaciones de valores de los pesos son las responsables del rendimiento de la red a la hora de realizar la tarea para la que ha sido diseñada. Una red neuronal *aprende* cuando, a través del ajuste de estos valores, la red obtiene un rendimiento aceptable. Este proceso de ajuste se denomina entrenamiento.

El descenso de gradiente estocástico o SGD, es el algoritmo de optimización más popular en el entrenamiento de redes neuronales. Este algoritmo tiene como objetivo encontrar un mínimo local de una función diferenciable, conocida como función de coste. Partiendo de la configuración inicial de los pesos de la red neuronal, SGD ajusta los parámetros de la red aplicando la función

$$f(w) = w - \eta \nabla l_i(w)$$

donde w es el vector de parámetros actuales, η es la tasa de aprendizaje, y $l_i(w)$ es la función de coste para el dato i . En la Figura 2.4 puede observarse como utilizando SGD se alcanza un mínimo en la función de coste, entrenando así una red.

La tasa de aprendizaje η es un coeficiente que permite ajustar la influencia del gradiente en el cálculo de parámetros. A medida que avanza el proceso el aprendizaje, es común reducir este valor para evitar el fenómeno conocido como *overshooting*, en el que un valor demasiado alto impide alcanzar el mínimo local (Figura 2.5).

Para reducir el número de pasos de cada iteración, las implementaciones de SGD dividen el conjunto de datos de entrenamiento en lotes y calculan la función de coste para cada lote en cada paso del algoritmo. Esta variante del SGD se conoce como descenso de gradiente mini-lote.

El proceso de entrenamiento, al igual que el de inferencia, requiere una gran capacidad de computación. Inicialmente, las redes neuronales se entrenaban con procesadores de propósito general. Más adelante, con la introducción de la plataforma CUDA, se comienzan a utilizar tarjetas gráficas o GPUs para esta tarea debido a su alta capacidad para realizar operaciones matemáticas en paralelo. Pese a que hoy en día existen procesadores específicos para el entrenamiento y ejecución de redes

neuronales (como por ejemplo, las unidades de procesamiento tensorial o TPUs), las GPUs continúan siendo una de las principales herramientas para esta tarea.

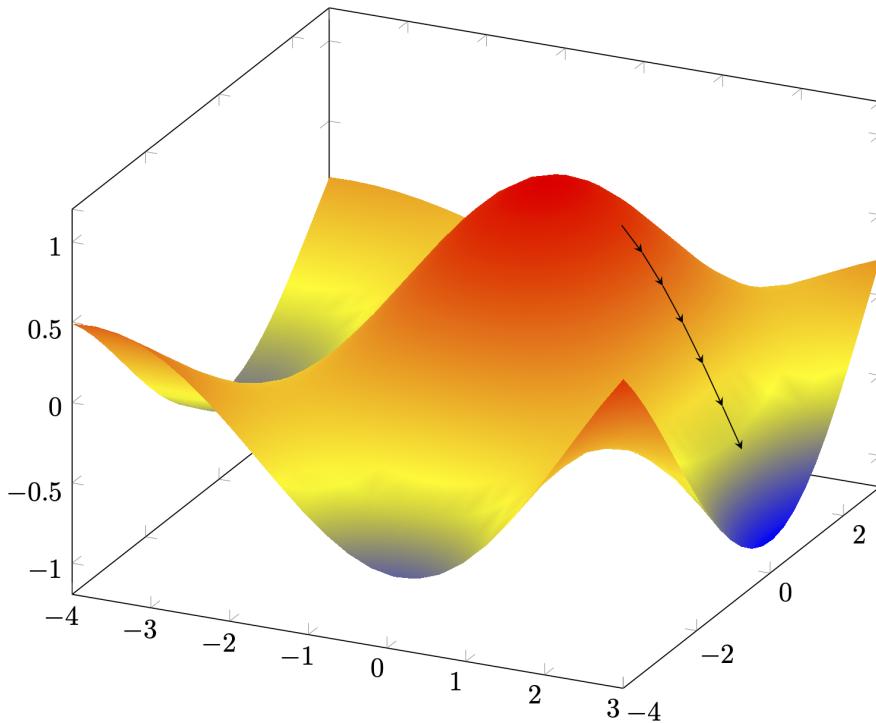


Figura 2.4: Ilustración del descenso de gradiente estocástico.

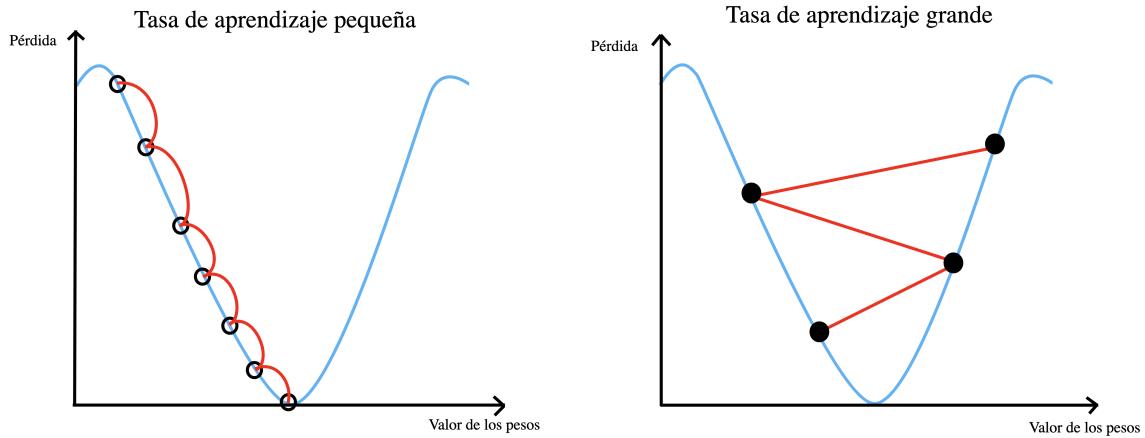


Figura 2.5: Efecto del *overshooting* en el SGD.

2.3. Redes convolucionales

El reconocimiento de patrones en imágenes es una tarea complicada para las redes neuronales tradicionales debido a la cantidad de variabilidad en la información de entrada y por el denominado efecto Hughes (Oommen et al., 2008). Este efecto,

también conocido como la maldición de la dimensionalidad, describe el aumento en complejidad que surge al procesar datos en espacios de alta dimensionalidad, como las imágenes.

Para hacer frente a esta dificultad se introducen las redes neuronales convolucionales o CNNs. Basadas en el neocognitron de Fukushima (1980), las CNNs consiguen extraer patrones de imágenes con una gran versatilidad. Esta clase de redes neuronales introducen dos nuevos tipos de capas al comienzo de la red: la capas convolucionales y las capas de agrupación o *pooling*. El resto de capas, propias de una red neuronal tradicional, se denominan capas *Fully Connected* o FC.

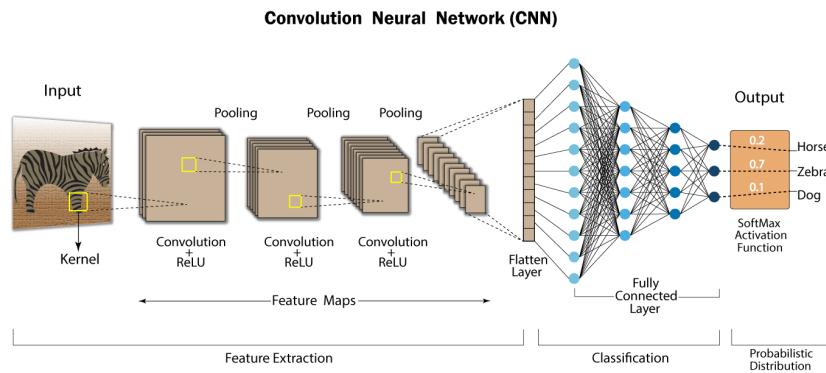


Figura 2.6: Arquitectura típica de una CNN (Koushik, 2023).

Las capas convolucionales aplican filtros convolucionales con el objetivo de reconocer patrones en una imagen. Estos filtros, como el de la Figura 2.7, realizan un producto escalar sobre grupos de píxeles adyacentes con una matriz llamada filtro o *kernel*, produciendo un mapa de características representados en la Figura 2.8. Estos mapas de características reconocen rasgos específicos, como una línea recta, o un círculo, por ejemplo. La concatenación de varias capas convolucionales permite reconocer patrones más complejos, como la presencia de un perro en una imagen.

Tras aplicar capas convolucionales, es común aplicar una capa de agrupación para reducir la dimensionalidad de los mapas de características preservando las características generales. Esta capa combina grupos de píxeles adyacentes utilizando una función de agrupación, permitiendo reducir el número de parámetros. Esto permite que las capas FC tengan una complejidad mucho menor, evitando así el efecto Hughes.

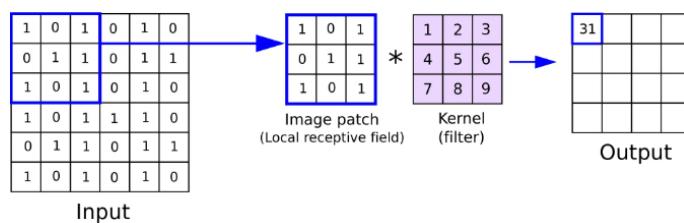


Figura 2.7: Aplicación de un filtro convolucional sobre una entrada (Akhtar et al., 2022).

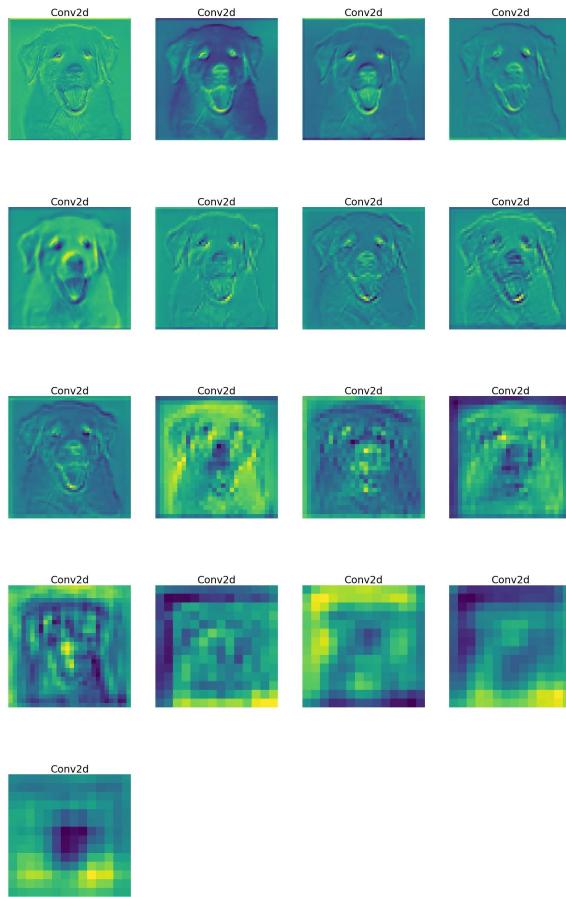


Figura 2.8: Mapas de características de una CNN para reconocer perros (vaishnav, 2020).

Las capas de agrupación se dividen en varios tipos según a la función utilizada, siendo tres los más comunes: las capas Min Pooling, Max Pooling y Average Pooling, que preservan el valor mínimo, máximo o realizan una media del grupo, respectivamente (Figuras 2.9 a 2.11).

Tanto los filtros de las convoluciones como las funciones de agregación de las capas de agrupación se comportan como ventanas deslizantes que recorren la matriz de píxeles aplicando la función correspondiente. Si bien se puede hacer secuencialmente este recorrido para cada píxel, es posible definir un desplazamiento fijo de esta ventana después de cada función de agregación. A este salto o desplazamiento se le llama *stride* y repercute directamente en el tamaño de salida de la capa a la que se aplica.

Por otra parte, al realizar operaciones como la convolución se puede no preservar el tamaño original o el deseado. Para abordar este problema utilizamos el relleno o *padding*, que añade los bordes necesarios para preservar la dimensión requerida utilizando ceros o los valores de los píxeles vecinos.

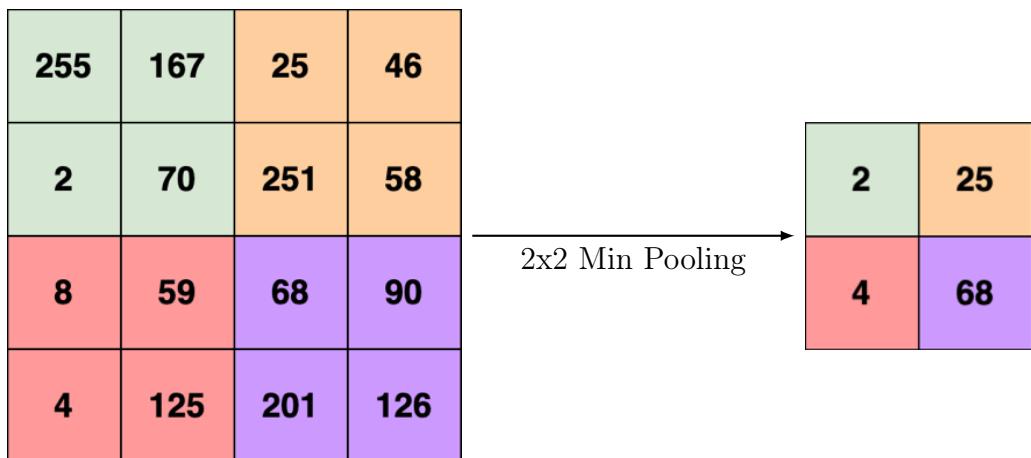


Figura 2.9: Ejemplo de Min Pooling (Soto, 2022).

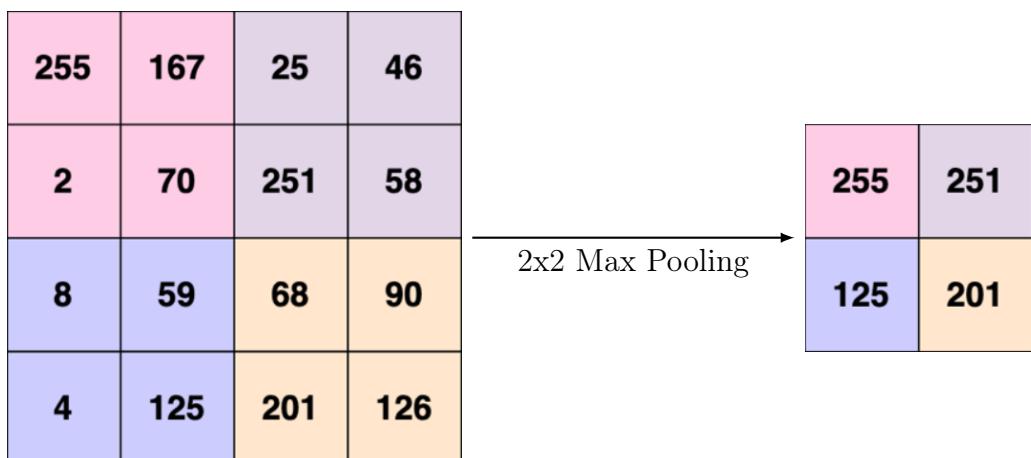


Figura 2.10: Ejemplo de Max Pooling (Soto, 2022).

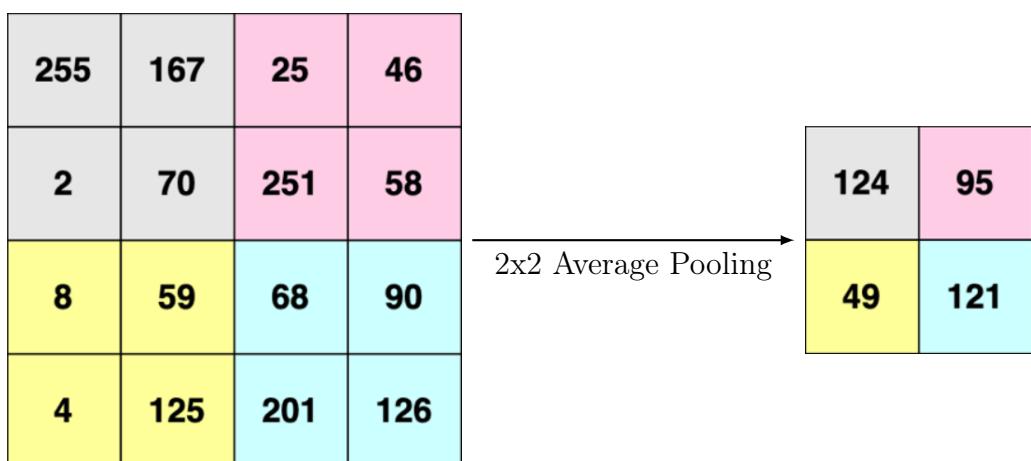


Figura 2.11: Ejemplo de Average Pooling (Soto, 2022).

Capítulo 3

Estado del Arte

Desde el principio de los años setenta, se han realizado numerosos estudios sobre las expresiones faciales de los seres humanos. Poco después comienza a haber un interés por la automatización de su reconocimiento. Desde entonces, los métodos utilizados para resolver esta tarea han ido evolucionando en sincronía con los avances en la capacidad computacional, siempre con la misma estrategia: la clasificación en base al reconocimiento de patrones faciales. Esta evolución se puede clasificar en tres grandes etapas: metodologías clásicas, la introducción del *machine learning* y las redes neuronales convolucionales, y las arquitecturas mixtas.

3.1. Metodologías clásicas

Los primeros acercamientos al reconocimiento de expresiones faciales están basados en el *Facial Action Coding System* (FACS) (Ekman y Friesen, 1978). Este sistema surge por la necesidad de establecer un sistema objetivo y estandarizado para describir las expresiones faciales. En FACS, las expresiones se dividen en componentes unitarios de movimientos musculares, denominados unidades de acción o *Action Units* (AUs). En la Tabla 3.1 se pueden observar varios ejemplos de unidades de acción.

En total, la versión original de FACS define 46 unidades de acción diferentes. Las expresiones faciales complejas son el resultado de dos o más unidades de acción. Por ejemplo, una sonrisa es una combinación de dos unidades de acción: tiramiento labial esquimal (AU12) y levantamiento de mejillas (AU6).

Los primeros sistemas de reconocimiento de expresiones faciales basan su funcionamiento en este estándar. La gran mayoría se centran en el reconocimiento a través de vídeos, utilizando complicados modelos físicos tridimensionales como el mostrado en la Figura 3.1 para reconocer y sintetizar expresiones faciales (Lien et al., 1998). Una vez obtenidas las unidades de acción, se utiliza un clasificador para obtener una expresión facial. Finalmente, en algunos casos, se incluye una fase de postprocesado para corregir errores de clasificación.

El empleo de estas estrategias implica diversas limitaciones. En primer lugar, se requiere un control riguroso de las condiciones de captura de la imagen (la posición de la cabeza, la presencia de ruido ambiental, la calidad de la iluminación...).

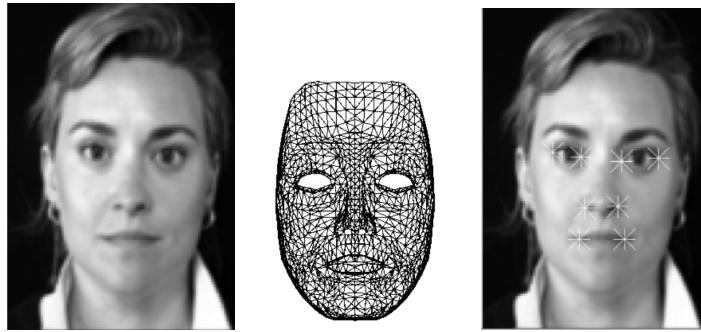


Figura 3.1: Aplicación de un modelo físico para extraer unidades de acción: imagen original, modelo geométrico, y extracción (Vadapalli, 2011).

Por otro lado, aunque el FACS proporciona una base sólida para la descripción de expresiones faciales, su capacidad de reconocimiento es limitada para expresiones sutiles o de mayor complejidad. Por ejemplo, ciertas emociones pueden ser ambiguas o variar considerablemente entre individuos, lo que dificulta su interpretación precisa mediante un enfoque puramente basado en unidades de acción musculares.

El proceso de ajuste inicial del modelo geométrico a la imagen a menudo requiere intervención humana (Lien et al., 1998), lo que implica una inversión considerable de tiempo y recursos. Aunque se han desarrollado algoritmos de flujo óptico para adaptar el modelo a los movimientos del individuo, esta etapa puede seguir siendo un cuello de botella en la implementación práctica de estos sistemas. Asimismo, la precisión del reconocimiento puede degradarse significativamente en entornos con condiciones variables o poco controladas, lo que limita su aplicabilidad en escenarios reales.

En resumen, aunque el FACS ha sentado las bases para el reconocimiento de expresiones faciales, su aplicación implica desafíos significativos relacionados con la precisión, la escalabilidad y la adaptabilidad a entornos variables.

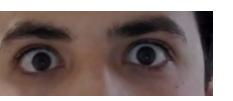
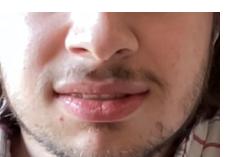
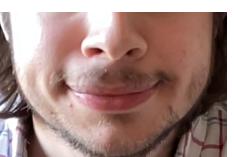
Unidades de acción superior			
AU 1	AU 2	AU 4	AU 5
			
Levantamiento interior de ceja	Levantamiento exterior de ceja (unilateral, lado derecho)	Bajar cejas	Levantamiento del párpado superior
AU 6	AU 7	AU 41	AU 42
			
Levantamiento de mejillas	Apretar párpado(s)	Bajada de párpados	Contracción retinal
AU 43	AU 44	AU 45	AU 46
			
Ojos cerrados	Recolector retinal	Parpadeo	Guiño
Unidades de acción inferior			
AU 9	AU 10	AU 11	AU 12
			
Arrugar la nariz	Levantamiento del labio superior	Profundidad nasolabial	Tiramiento labial esquinal
AU 13	AU 14	AU 15	AU 16
			
Tiramiento labial frontal	Hoyuelo facial	Depresión labial esquinal	Depresión labial frontal
AU 17	AU 18	AU 20	AU 22
			
Levantamiento de barbilla	Arruga labial	Apretar los labios	Embudo labial

Tabla 3.1: Unidades de acción principales.

3.2. Redes neuronales convolucionales

Los trabajos de Fasel (2002) y Matsugu et al. (2003) son posiblemente las primeras propuestas de una red neuronal convolucional para el reconocimiento de expresiones faciales. Esta iniciativa resuelve uno de los principales problemas de las técnicas anteriores: la necesidad de controlar las condiciones de la imagen o vídeo a procesar.

Una de las primeras arquitecturas de gran éxito es AlexNet, propuesta por Krizhevsky et al. (2017) (Figura 3.2). Esta arquitectura logró resultados innovadores en el reconocimiento de objetos en imágenes naturales. Esta arquitectura demuestra la importancia de la profundidad de la red para mejorar su rendimiento. Para poder extraer patrones más complejos, AlexNet introduce varias capas convolucionales con filtros de gran tamaño (concretamente, 5x5 y 11x11).

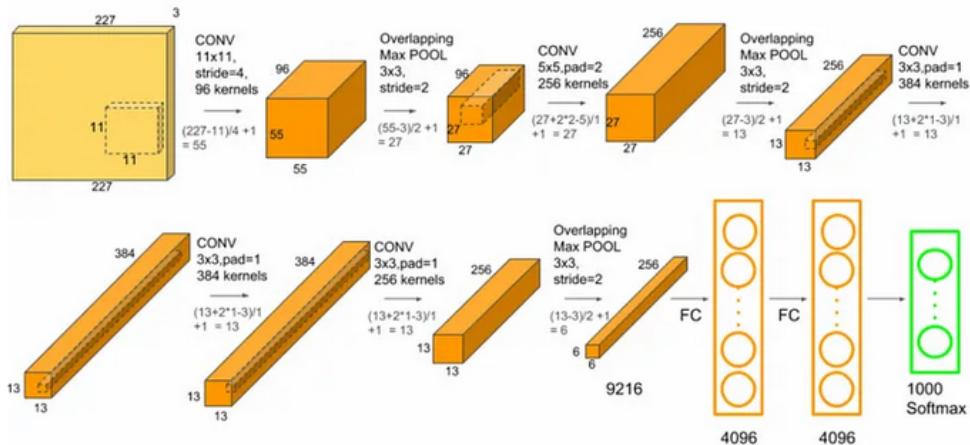


Figura 3.2: Arquitectura AlexNet (Nabil, 2023).

El aumento de profundidad en la red, pese a dar buenos resultados, provoca un alto sobreaprendizaje. Srivastava et al. (2014b) consigue reducir considerablemente este problema con la introducción de las capas *Dropout*. Estas capas, situadas entre capas FC, ignoran intermitentemente las salidas de las neuronas conectadas a ella.

Por otro lado, el uso de la función de activación ReLU mejoró también los resultados al disminuir el problema del gradiente evanescente (Ide y Kurita, 2017). Este fenómeno ocurre cuando, durante el entrenamiento, los gradientes que se utilizan para actualizar la red se vuelven extremadamente pequeños o “desaparecen” a medida que se retropropagan desde las capas de salida a las capas anteriores.

El *Visual Geometry Group* (VGG) de la Universidad de Oxford, propuso una nueva arquitectura basada en la simplicidad, en la utilización de filtros pequeños (3 x 3) y en una muy profunda pila de capas (Figura 3.3). Esta arquitectura fue denominada VGG (Simonyan y Zisserman, 2015) y estableció una nueva tendencia hacia estos filtros al demostrar que tenían la misma influencia que filtros mayores,

pero disminuyendo en gran medida los parámetros necesarios.

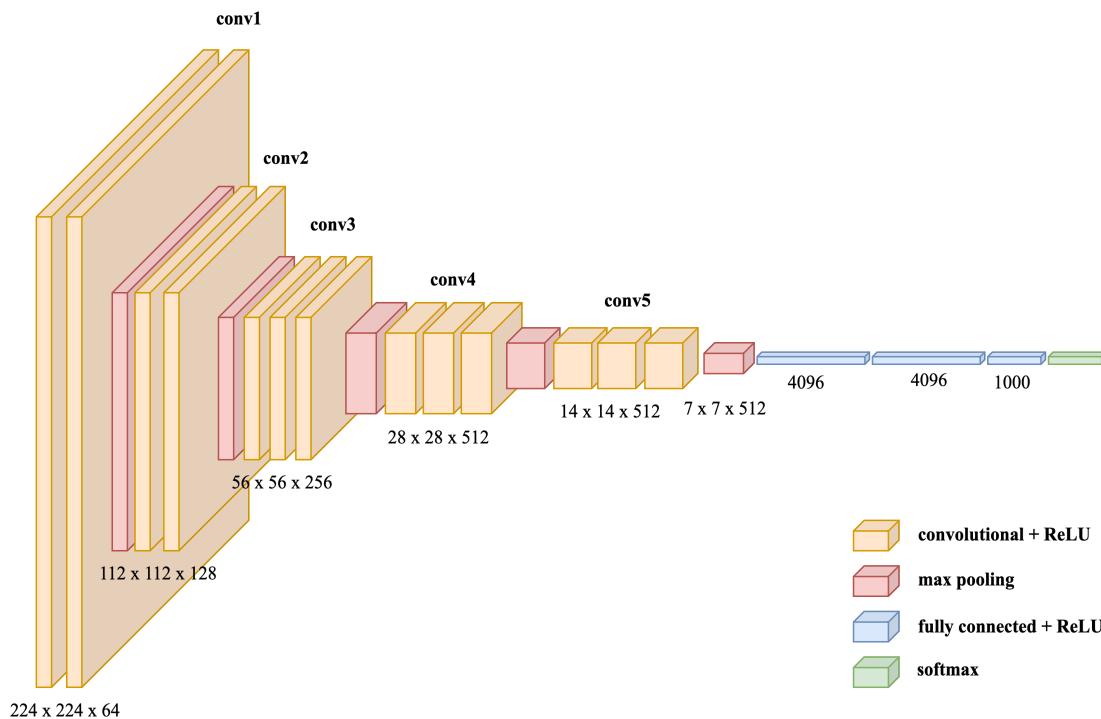


Figura 3.3: Arquitectura VGGNet-16.

Otra arquitectura de gran éxito es GoogLeNet, la ganadora del concurso ILSVRC de 2014 (Szegedy et al., 2015). Esta arquitectura introdujo un nuevo enfoque empleando convoluciones de diferentes filtros de forma paralela (Figura 3.4). Este enfoque permite capturar las características de forma eficaz y a mayor escala, ayudando a una mejor generalización. Además, implementa una capa de agregación media global en vez de una capa FC, permitiendo reducir los parámetros de 40 millones a únicamente 5.

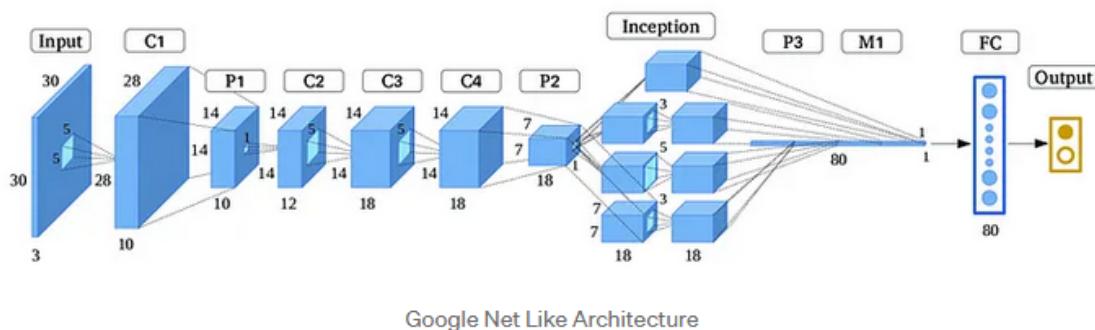


Figura 3.4: Arquitectura GoogLeNet (Nabil, 2023).

Aunque la implementación de la función de activación ReLU parecía haber resuelto el problema del gradiente evanescente, este vuelve a aparecer en redes de gran profundidad. Para resolverlo, He et al. (2016) introduce ResNet. En esta arquitectura se añaden conexiones que sirven como atajos en el flujo de la red, permitiendo que se puedan evitar una o mas capas (Figura 3.5). Esta idea consigue que el gradiente pueda llegar a las capas iniciales, eliminando así el problema del gradiente evanescente en modelos de hasta 100 capas convolucionales.

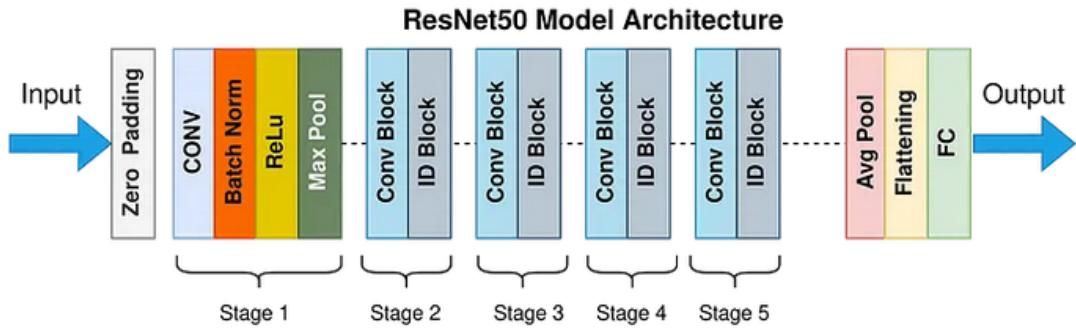


Figura 3.5: Arquitectura ResNet (Nabil, 2023).

Aunque muchas de las arquitecturas discutidas en esta sección no fueron diseñadas específicamente para el reconocimiento de emociones faciales, demostraron ser altamente efectivas para esta tarea debido a sus innovaciones y capacidades avanzadas de extracción de características. Sin embargo, las CNNs puras presentan limitaciones significativas que se resuelven con arquitecturas más complejas.

3.3. Arquitecturas mixtas

El reconocimiento de emociones en vídeos presenta un desafío único debido a la naturaleza temporal de los datos. Las CNN tradicionales, si bien son poderosas en la extracción de características visuales, carecen de memoria a largo plazo y procesan cada fotograma de forma independiente. Esta limitación impide que capturen la evolución temporal de las expresiones faciales a lo largo del vídeo.

Estas deficiencias motivaron el desarrollo de arquitecturas híbridas que combinan CNN con otros modelos, como las máquinas de soporte vectorial (SVM) o las redes neuronales de memoria a largo plazo (LSTM) (Nie et al., 2020; Sun et al., 2016; Singh, 2022). Estas arquitecturas buscan aprovechar lo mejor de ambos mundos: la potente capacidad de extracción de características de las CNN y la habilidad de los modelos secuenciales para manejar información temporal, ofreciendo así una solución más robusta y eficaz para el reconocimiento de emociones en vídeos.

Las arquitecturas CNN+LSTM logran capturar la dinámica de las expresiones faciales a lo largo del tiempo, dando una comprensión más completa de las emociones expresadas (Figura 3.6). Así, este tipo de modelos entienden cómo las emociones evolucionan y se interconectan a lo largo de la duración del vídeo.

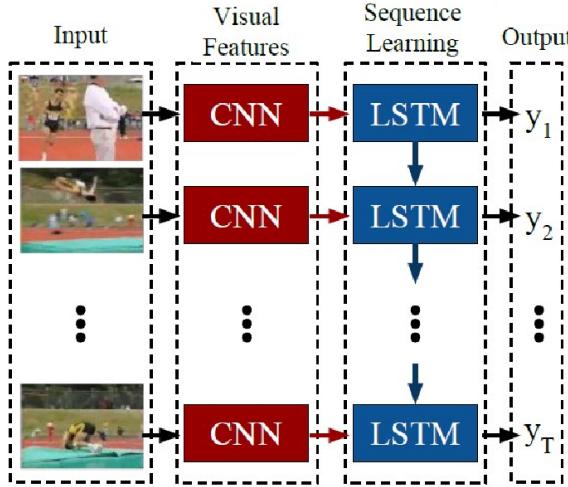


Figura 3.6: Ejemplo de CNN y LSTM (Akhtar et al., 2022).

La arquitectura DDAMFN de Zhang et al. (2023) presenta una estructura característica compuesta por dos fases: MFN y DDA (Figura 3.7). La fase MFN, inspirada en la arquitectura de las redes neuronales convolucionales, se encarga de generar mapas iniciales de características a partir de la imagen de entrada. Sin embargo, en lugar de emplear convoluciones convencionales, la arquitectura hace uso de convoluciones en profundidad o *depthwise*. Estas convoluciones son una técnica especializada que permite reducir drásticamente la cantidad de parámetros del modelo, mejorando la eficiencia computacional y reduciendo el riesgo de sobreaprendizaje. Por otro lado, la fase DDA se encarga de procesar los mapas de características generados por la MFN para producir mapas de atención tanto vertical como horizontalmente. Estos mapas son esenciales para la identificación de las regiones de la imagen más relevantes para determinar la emoción facial. Finalmente, los mapas de atención se introducen en una capa FC que determina la emoción facial expresada en la imagen.

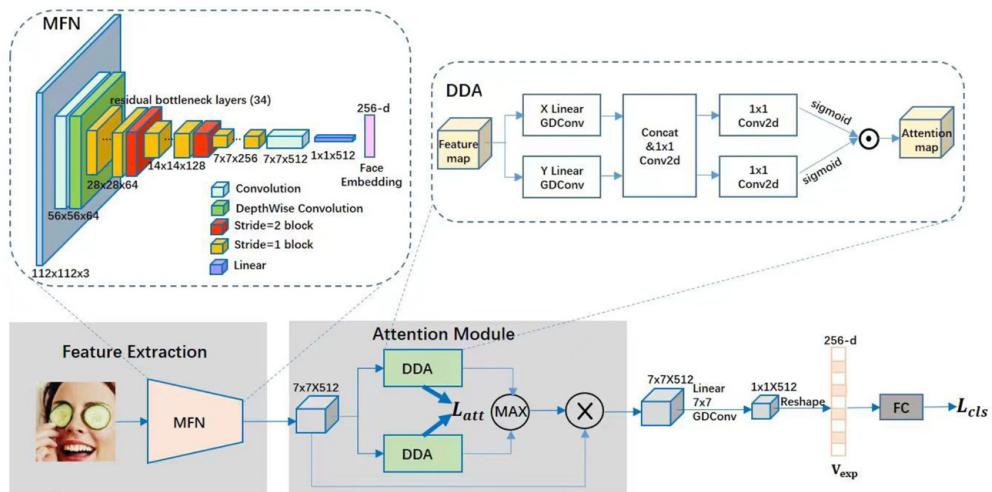


Figura 3.7: Estructura de la arquitectura DDAMFN (Zhang et al., 2023).

POSTER es otra arquitectura relevante en el campo del FER (Zheng et al., 2023). Esta arquitectura combina la potencia de las CNN para extraer características visuales de la imagen con la capacidad de detectar puntos característicos faciales (Figura 3.8). Esta combinación de características visuales y puntos faciales se fusiona en tres matrices, que luego se utilizan para determinar la emoción presente en la imagen.

Por último, modelos como BoVW combinan CNNs con SVMs (Georgescu et al., 2019). Este modelo aprovecha las capas convolucionales de CNNs previamente entrenadas (concretamente, VGG-13, VGG-f y VGG-face), cuyas salidas concatenan e introduce en un modelo SVM local para el reconocimiento de emociones (Figura 3.9).

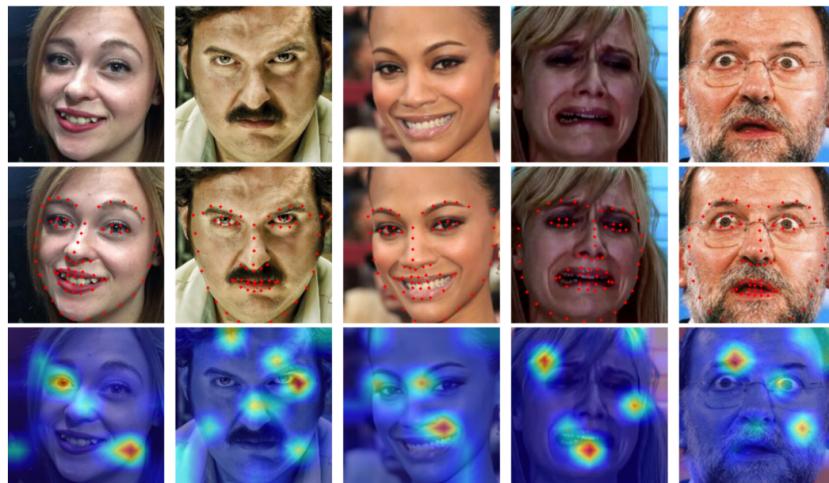


Figura 3.8: Extracción de características y puntos clave de una imagen (ju Mao et al., 2023).

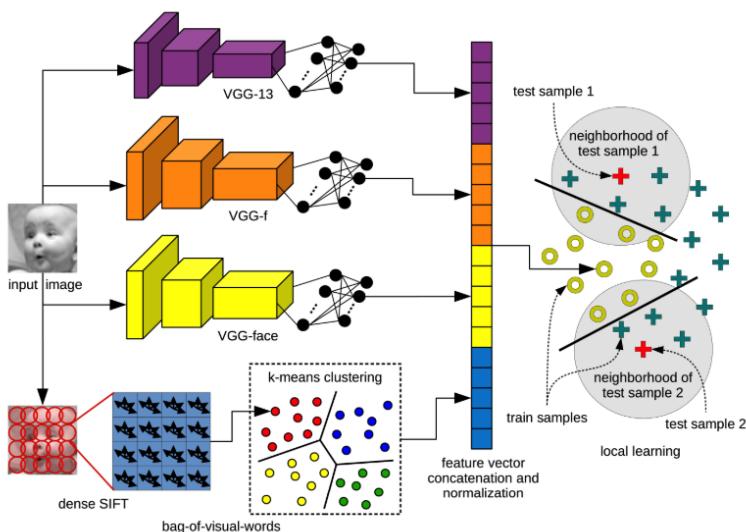


Figura 3.9: Estructura del modelo BoVW (Georgescu et al., 2019).

Capítulo 4

Nuestra propuesta

El rendimiento de las arquitecturas de tipo VGG ha demostrado la efectividad del uso de filtros pequeños y un alto número de capas convolucionales para el reconocimiento de objetos en imágenes. En concreto, el modelo VGGNet-16 alcanza un 92.7 % en el Top-5 Accuracy Test de ImageNet, en el que la red debe clasificar imágenes en 1000 clases diferentes y el resultado correcto ha de estar en las cinco categorías de mayor probabilidad (Simonyan y Zisserman, 2015).

Esta red está formada por dos capas convolucionales con 64 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas convolucionales con 128 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 256 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 512 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 512 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas FC con 4096 neuronas, y finalmente una capa FC con 1000 neuronas (Figura 4.6). El número de parámetros resultante es de 138 millones.

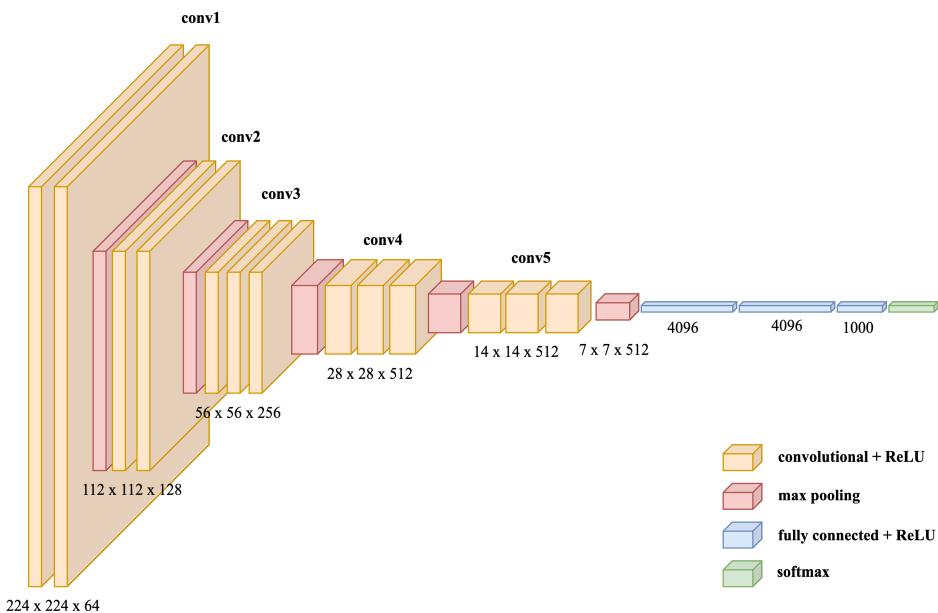


Figura 4.1: Arquitectura VGGNet-16.

Para este trabajo proponemos cinco arquitecturas distintas: VGG10a, VGG10b, VGG12, VGG14 y VGG17. Todas ellas reciben como entrada una imagen de dimensiones 128x128 en escala de grises. A diferencia de VGGNet, se utiliza la función de activación LeakyReLU, que permite evitar el problema de ReLU en el que ciertas conexiones “mueren” al asignarles un peso nulo (para una revisión más detallada de este problema, consultar Versloot (2019)).

4.1. VGG10a

VGG10a es una red neuronal con 8 capas convolucionales y 2 capas FC. Su estructura consiste en dos capas convolucionales con 32 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas convolucionales con 64 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas convolucionales con 128 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas convolucionales con 256 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, una capa FC con 512 neuronas y otra capa FC con 7 neuronas (Figura 4.2). El número de parámetros resultante es de 9.5 millones.

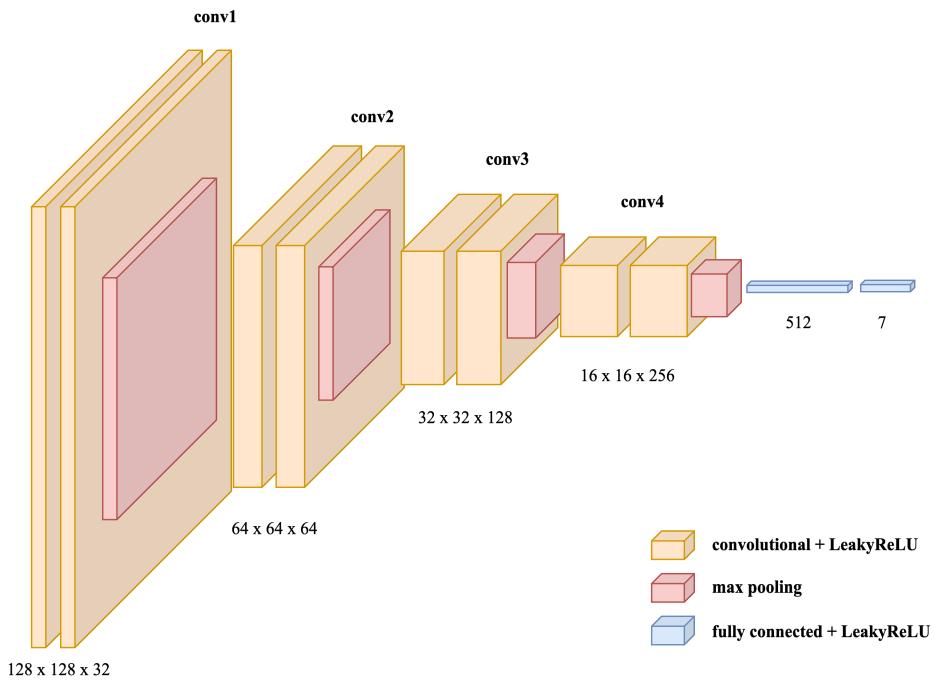


Figura 4.2: Arquitectura VGG10a.

4.2. VGG10b

VGG10b es una red neuronal con 8 capas convolucionales y 2 capas FC. Su estructura consiste en dos capas convolucionales con 64 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas convolucionales con 128 filtros 3x3, una

capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas convolucionales con 256 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas convolucionales con 512 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, una capa FC con 512 neuronas y otra capa FC con 7 neuronas (Figura 4.3). El número de parámetros resultante es de 21.4 millones.

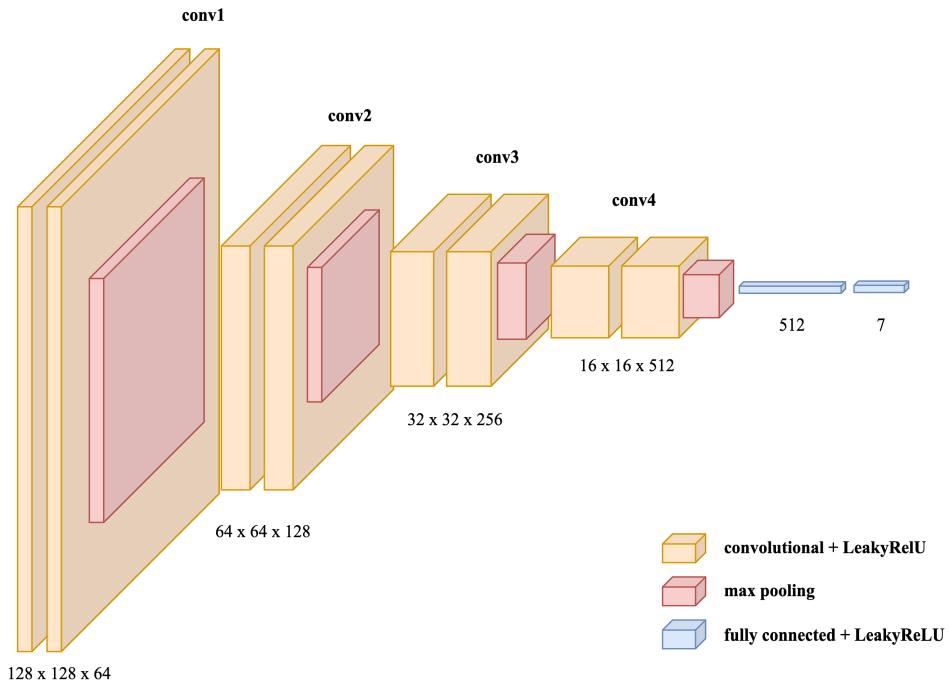


Figura 4.3: Arquitectura VGG10b.

4.3. VGG12

VGG12 es una red neuronal con 10 capas convolucionales y 2 capas FC. Su estructura consiste en dos capas convolucionales con 64 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, dos capas convolucionales con 128 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 256 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 512 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, una capa FC con 512 neuronas y otra capa FC con 7 neuronas (Figura 4.4). El número de parámetros resultante es de 23.1 millones.

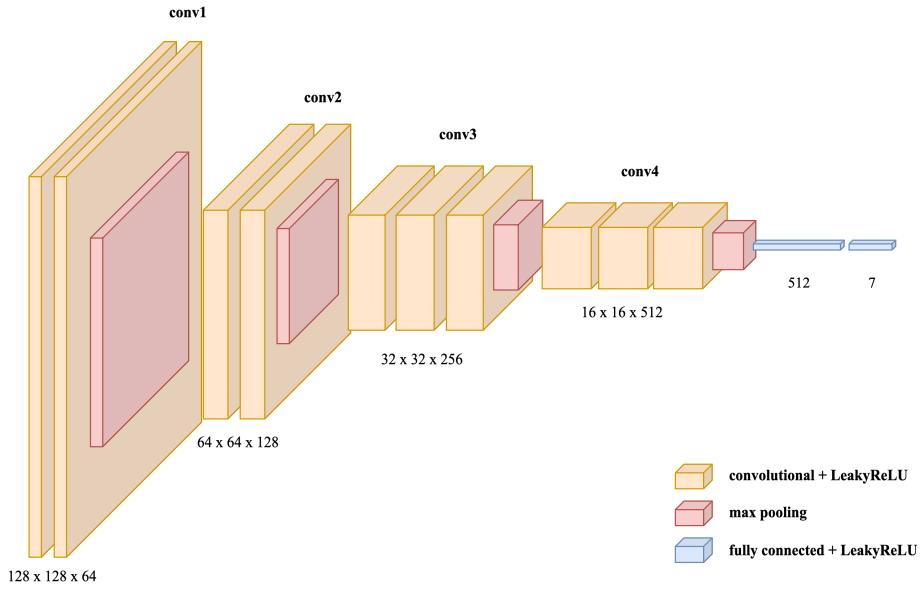


Figura 4.4: Arquitectura VGG12.

4.4. VGG14

VGG14 es una red neuronal con 12 capas convolucionales y 2 capas FC. Su estructura consiste en tres capas convolucionales con 64 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 128 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 256 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 512 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, una capa FC con 512 neuronas y otra capa FC con 7 neuronas (Figura 4.5). El número de parámetros resultante es de 23.3 millones.

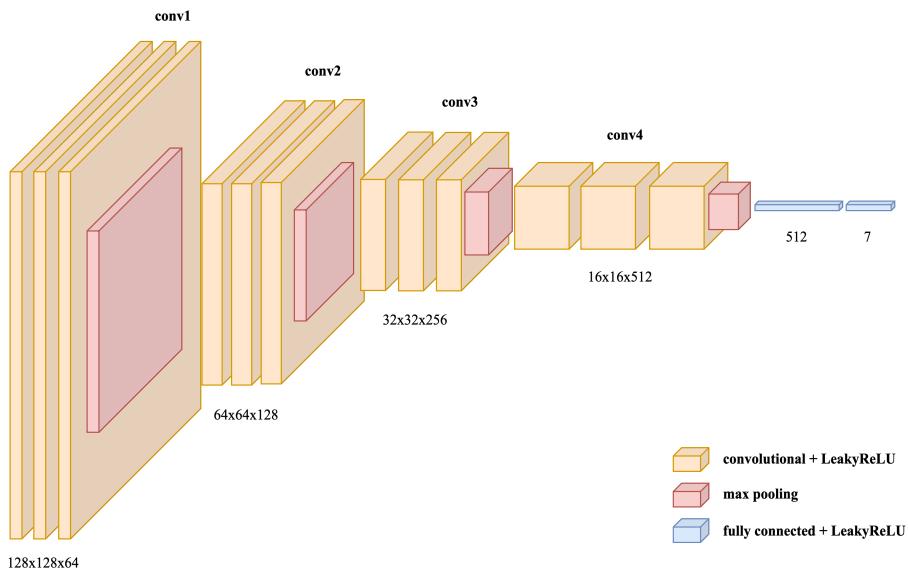


Figura 4.5: Arquitectura VGG14.

4.5. VGG17

VGG17 es una red neuronal con 15 capas convolucionales y 2 capas FC. Su estructura consiste en tres capas convolucionales con 64 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 128 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 256 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 512 filtros 3x3, una capa *Max Pooling* de tamaño 2x2 y *stride* 1, tres capas convolucionales con 512 filtros 3x3, una capa FC con 512 neuronas y otra capa FC con 7 neuronas (Figura 4.6). El número de parámetros resultante es de 28 millones.

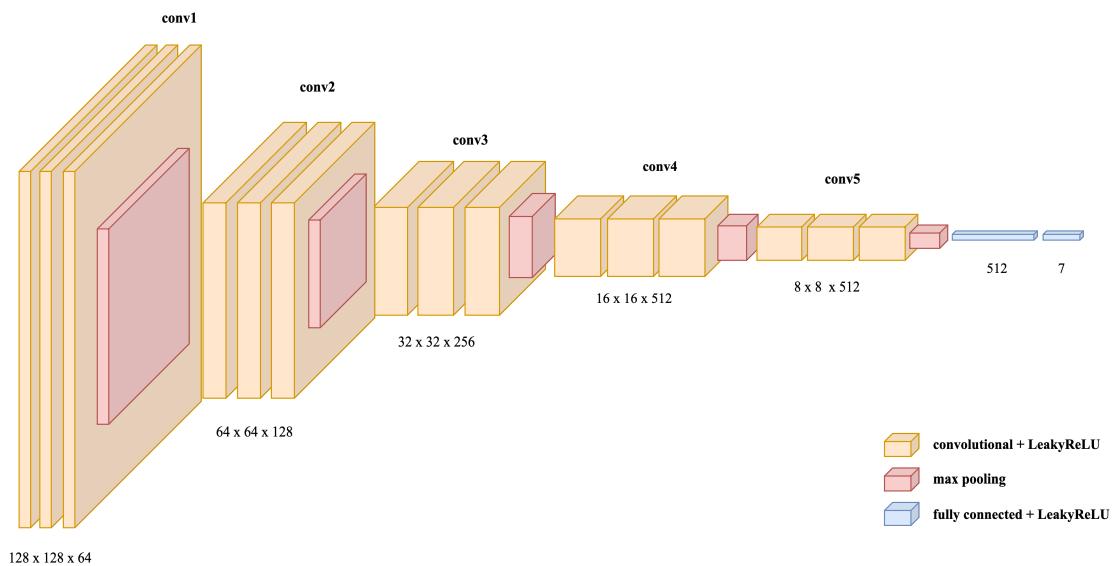


Figura 4.6: Arquitectura VGG17.

Capítulo 5

Experimentación

Este capítulo recoge todos los detalles del proceso de entrenamiento que hemos seguido para entrenar los diferentes modelos propuestos. Todos los experimentos descritos a continuación se realizaron con entrenamientos de 50 *epochs* y 64 imágenes por lote en cuatro GPUs NVIDIA V100 a través de la plataforma *Google Cloud*.

5.1. Dataset

Para entrenar una red neuronal es necesario contar con un dataset. Es necesario contar con un conjunto de datos clasificados por individuos expertos que permita evaluar el rendimiento de la red neuronal, conocido como dataset.

Existen varios datasets de acceso público con los que entrenar un modelo de reconocimiento de expresiones faciales. Muchos de ellos, no obstante, son limitados. FER-2013, introducido en el concurso *Facial Expression Recognition Challenge* en la ICML 2013, es un dataset con 36000 imágenes de diversas expresiones faciales (Goodfellow et al., 2013). El Extended Cohn-Kanade Dataset o CK+ (Lucey et al., 2010) contiene 593 vídeos de 123 sujetos distintos mostrando diferentes expresiones faciales, clasificados en siete emociones. RAF-DB (Li y Deng, 2019) está formado por aproximadamente 30000 imágenes de individuos obtenidas de Internet, clasificadas en siete emociones.

Para el entrenamiento de la red neuronal propuesta decidimos utilizar AffectNet (Mollahosseini et al., 2019), una de las bases de datos más extensas de imágenes de individuos en entornos no controlados. Este conjunto cuenta con 456349 imágenes obtenidas de Internet utilizando palabras claves relacionadas con las emociones en seis idiomas distintos, y manualmente clasificadas por doce expertos en once categorías: neutral, felicidad, tristeza, sorpresa, miedo, rabia, asco, desprecio, ninguna, incierto y sin rostro. De ellas, el 69.89 % corresponde a una de las ocho primeras categorías (Tabla 5.1). Por ello, decidimos excluir el resto de categorías del conjunto de datos para el entrenamiento. En la Tabla 5.2 se pueden ver varios ejemplos de imágenes del dataset.

Seis de las siete emociones representadas en estas categorías son emociones universales (Ekman y Friesen, 1971), con la excepción de la categoría “desprecio”. Esta última emoción es también la menos representada, y sus expresiones faciales son

muy sutiles, compartiendo unidades de acción con otras de las expresiones propias de categorías diferentes. Por esta razón, y por motivos de consistencia con otros estudios similares (Zeng et al., 2018; Li et al., 2019; Chen et al., 2019; Wang et al., 2019), hemos decidido excluir esta categoría del entrenamiento. El subconjunto de prueba, que no está disponible al público, también queda excluido.

En total, las siete categorías restantes contienen 287402 imágenes, las cuales se dividen en dos subconjuntos: el subconjunto de entrenamiento, formado por 283902 imágenes; y el subconjunto de validación público, compuesto por 3500 imágenes.

Clase	Muestras	Porcentaje
Desprecio	5135	01.13 %
Felicidad	146198	32.04 %
Rabia	28130	06.17 %
Miedo	8191	01.79 %
Neutral	80276	17.59 %
Asco	5264	01.15 %
Sorpresa	16288	03.57 %
Tristeza	29487	06.46 %
Ninguna	35322	07.74 %
Incierto	13163	02.88 %
Sin rostro	88895	19.48 %

Tabla 5.1: Distribución de datos de AffectNet.

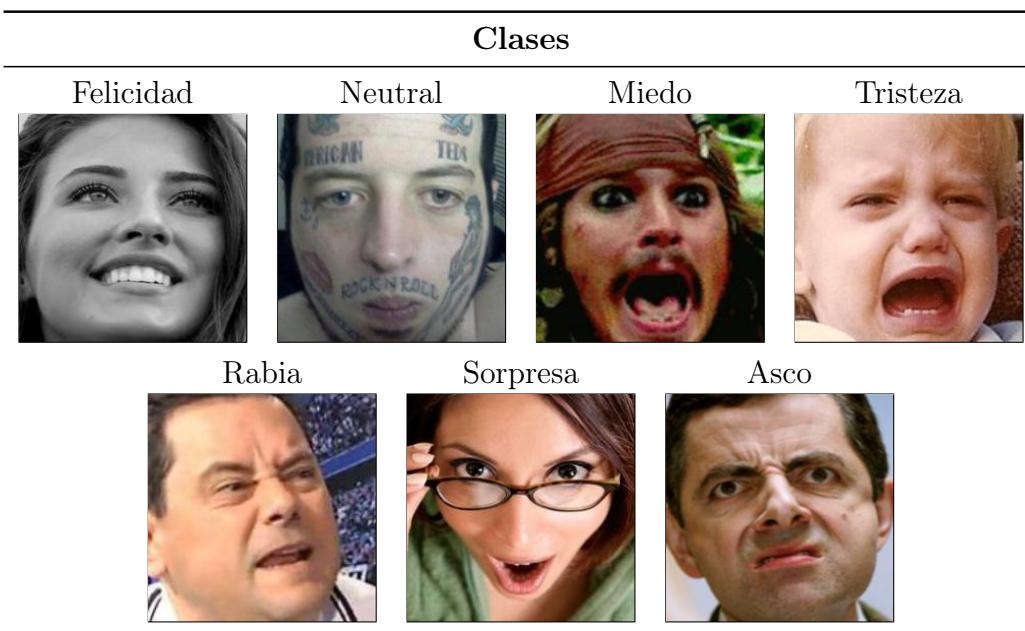


Tabla 5.2: Ejemplos de distintas clases de AffectNet.

5.2. Preprocesamiento

La preparación de los datos es uno de los pasos más importantes previos al entrenamiento de una red neuronal. En ocasiones, los datos de entrenamiento están incompletos o no tienen el formato apropiado. Estos requisitos varían notablemente en función de la tarea que la red debe realizar.

Las imágenes de AffectNet tienen una dimensión media de 425x425 píxeles y desviación estándar de 349x349 píxeles. Tras seleccionar la región de cada una de ellas con atributos faciales y escalar los datos, todas las imágenes pasan a ser de 224x224 píxeles. Todas ellas están en formato RGB, es decir, tienen tres canales, cada uno representando la presencia del color correspondiente (rojo, verde o azul) para cada píxel con un valor en el rango [0, 256].

De acuerdo con el trabajo de Yudin et al. (2020), la presencia de colores no aporta una ventaja significativa en el reconocimiento de expresiones faciales. Por esta razón, y para reducir el número de parámetros y los requisitos de memoria y capacidad de computación durante el entrenamiento, se convierten todas las imágenes a escala de grises (Figura 5.1).



Figura 5.1: Transformación a escala de grises.

El tamaño de las imágenes de entrada en una red convolucional es también un parámetro de gran importancia. Es intuitivo pensar que, a mayor tamaño de imagen, mayor detalle y, por tanto, mejores resultados (Figura 5.2). Sin embargo, tamaños muy grandes no solo incrementan considerablemente la complejidad del modelo y el tiempo de entrenamiento y ejecución, sino que el rendimiento puede también empeorar. Todas las arquitecturas de redes neuronales convolucionales tienen un tamaño de entrada preferente con el que dan mejores resultados (Richter et al., 2021). Este valor se puede determinar empíricamente. Para nuestro propósito reescalamos las imágenes a una resolución de 128x128, dimensiones con las que las imágenes son suficientemente descriptivas y los modelos no alcanzan un número de parámetros excesivamente alto.

La normalización de los valores de entrada es una práctica recomendable en las redes neuronales. Como los valores de los píxeles de una imagen ya están acotados en el rango [0, 256], la normalización no es estrictamente necesaria pero sigue siendo recomendable. La principal razón detrás de esta normalización es incrementarla velocidad de aprendizaje. Por otra parte, al ser el estándar en el campo, muchas de las herramientas e implementaciones esperan recibir valores en ese intervalo. Asimismo,



Figura 5.2: Ejemplo de tamaños: 225x225, 128x128 y 64x64.

al normalizar y lograr una representación más eficiente y compacta de los datos se usarán los recursos computacionales de una manera mucho mas eficiente. Por todo esto, decidimos normalizar los valores de los píxeles de las imágenes de entrada del modelo.

Recientes investigaciones han explorado diversas técnicas de preprocesamiento de imágenes, entre las que destacan, aparte de la normalización local, la normalización del contraste global (GCN) y la ecualización del histograma (Pitaloka et al., 2017). El método GCN, que consiste en restar el valor medio de cada píxel de la imagen y dividirlo por la desviación estándar, tiene como objetivo asegurar que las imágenes presenten un contraste uniforme, lo cual es crucial en aplicaciones como el reconocimiento de expresiones faciales. Esta práctica mejora la robustez de modelos de aprendizaje automático, como las redes neuronales convolucionales, ante variaciones en la iluminación y otros factores ambientales. Por otro lado, la ecualización del histograma busca mejorar el contraste redistribuyendo los valores de intensidad de los píxeles en la imagen, lo que puede resaltar características y detalles ocultos debido a un contraste original deficiente. Esta técnica resulta especialmente útil en áreas como el procesamiento de imágenes médicas o la visión por computadora, donde la claridad y la interpretación precisa de las imágenes son cruciales.

5.3. Optimización

El uso de un optimizador y función de coste apropiadas es crucial para acelerar el proceso de aprendizaje. Esta sección aborda las diferentes alternativas a utilizar.

5.3.1. Función de coste

En el descenso de gradiente estocástico, es necesario definir una función que determine el rendimiento del modelo. Esta función, llamada función de coste, es inversamente proporcional al número de aciertos del modelo con el conjunto de datos de entrenamiento; es decir, a mayor número de aciertos del modelo, menor valor será el de la función.

Sea x_i la i-ésima entrada del conjunto de datos de entrenamiento, y_i la clase a la que pertenece dicha entrada, $f(x_i)$ las probabilidades de que la entrada pertenezca

a cada clase predicha por una red neuronal, la función de coste $L(y_i, f(x_i))$ calcula la diferencia entre ambos valores.

Existen muchas funciones de coste con diferentes características. En este trabajo decidimos utilizar la entropía cruzada, una de las funciones más populares. Esta función calcula el número de bits necesario para representar la probabilidad de la clase correcta de utilizando las probabilidades del modelo (para una revisión más detallada, consultar Brownlee (2020)).

5.3.2. Optimizadores de gradiente

Al igual que con las funciones de coste, existen diferentes implementaciones del descenso de gradiente estocástico.

Adam es uno de los optimizadores mas utilizados en el contexto de redes neuronales convolucionales y problemas derivados de reconocimiento facial. Basado en otros optimizadores como AdaGrad y RMSProp, Adam es un optimizador eficiente que se adapta a gradientes dispersos, tiene un número pequeño de hiperparámetros con una interpretación muy intuitiva, y da buenos resultados con conjuntos de datos amplios (Kingma y Ba, 2017). En comparación con otros optimizadores, Adam tiende a converger en poco tiempo (Figura 5.3). Sin embargo, una limitación de Adam es el uso de una tasa de aprendizaje fija durante todo el entrenamiento. Tradicionalmente, este problema se soluciona utilizando un planificador: un sistema que reduzca el valor de la tasa de aprendizaje cuando se cumplan una o varias condiciones, como que se hayan cumplido un número de iteraciones, la función de coste no haya cambiado en una cantidad de iteraciones, etc.

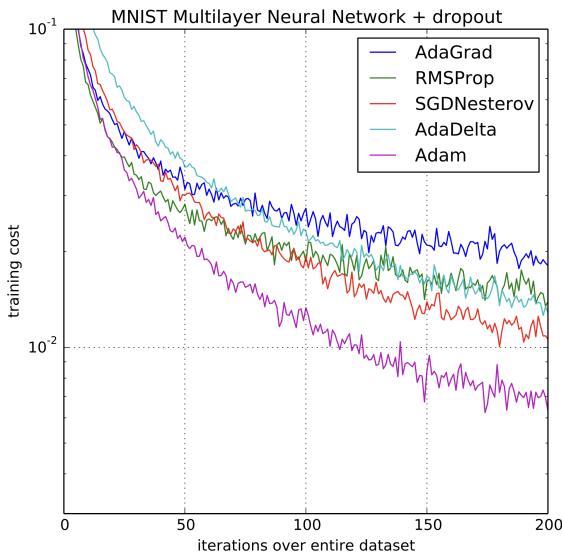


Figura 5.3: Evolución de la función de coste con diferentes optimizadores en una red neuronal multicapa MNIST (Sepúlveda, 2023).

El optimizador Schedule-free Adam de Defazio et al. (2024) es un novedoso optimizador que elimina la necesidad de ajustar manualmente la tasa de aprendizaje. Por esta razón, decidimos utilizar este optimizador.

5.4. Estrategias de aprendizaje

Uno de los desafíos más comunes al entrenar una red neuronal es el sobreaprendizaje u *overfitting*. Este fenómeno ocurre cuando el modelo aprende los datos de entrenamiento demasiado bien, hasta el punto de memorizarlos en vez de extraer patrones generales. Como resultado, el modelo tiene una precisión muy alta con los datos de entrenamiento, pero falla al replicar estos resultados con datos no vistos anteriormente. En la Figura 5.4 se puede observar como dos modelos, uno con sobreaprendizaje y otro sin él, clasifican datos en dos categorías distintas.

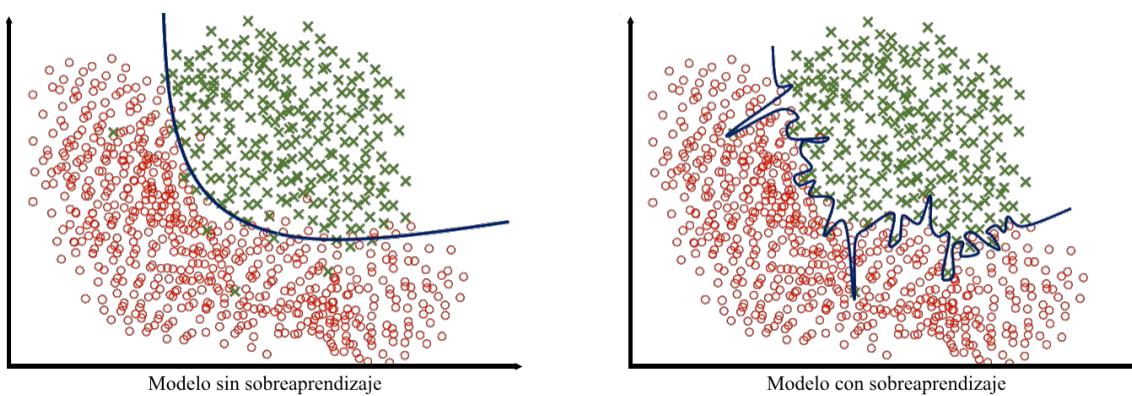


Figura 5.4: Efecto del sobreaprendizaje en un modelo.

Existen varias causas por las que puede surgir este problema, generalmente clasificadas en tres categorías: (i) tamaño inapropiado y/o ruido en el conjunto de datos de entrenamiento: utilizar un conjunto de datos de entrenamiento demasiado pequeño, con una notable diferencia en la representación de dos clases, o con mucho ruido. En esta situación, es probable que el modelo aprenda el ruido, y luego lo utilice para clasificar datos; (ii) complejidad del modelo: la cantidad de parámetros de un modelo permite captar diferentes patrones y relaciones en el conjunto de entrenamiento. Cuando este número es demasiado alto, el modelo puede continuar aumentando su precisión, a cambio de ser cada vez más inconsistente; y (iii) tiempo excesivo de entrenamiento: cuando un modelo es entrenado demasiado tiempo sobre el mismo conjunto de datos, éste tiende al sobreaprendizaje.

Las distintas estrategias explicadas a continuación servirán para evitar el problema del sobreaprendizaje.

5.4.1. Coste ponderado

Cuando se entrena un modelo con un conjunto de datos desequilibrado, la red neuronal tiende a reconocer las clases más representadas, y suele equivocarse con el resto. Por esta razón, es muy importante que el modelo se entrene con la misma cantidad de imágenes de todas las clases. Como se puede observar en la Figura 5.5, AffectNet es un conjunto de datos con una distribución heterogénea de cola larga; es decir, hay una gran diferencia entre las clases con mayor representación y las de menor.

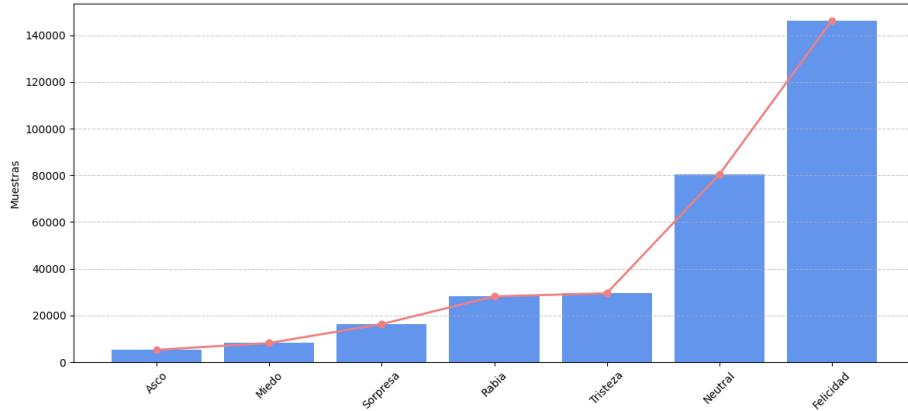


Figura 5.5: Distribución de clases en AffectNet.

Para solventar este problema, el coste ponderado aplica un coeficiente w_i , distinto para cada clase, en la función de coste. Este coeficiente, inversamente proporcional a la frecuencia de cada clase, aumenta el coste de los fallos de clasificación de las clases minoritarias, y se puede obtener con la ecuación

$$w_i = \frac{N}{N_i}$$

donde w_i es el coeficiente para la clase i , N es el número total de imágenes en el conjunto de datos, y N_i es el número de imágenes de la clase i . En la Tabla 5.3 están recogidos los coeficientes utilizados.

Clase	Muestras	Coeficientes de coste
Felicidad	134415	02.112
Neutral	74874	03.792
Miedo	6378	44.513
Tristeza	25459	11.151
Rabia	24882	11.410
Sorpresa	14090	20.150
Asco	5264	54.210

Tabla 5.3: Asignación de coeficientes de coste para cada clase.

5.4.2. Aumento de datos

En cada iteración del entrenamiento, el modelo aprende utilizando todas las imágenes del conjunto de datos, lo cual provoca el sobreaprendizaje. La técnica de aumento de datos evita este problema aplicando transformaciones aleatorias o controladas a los datos de entrada originales para crear nuevas muestras que son variaciones realistas de las muestras existentes. Existen muchos tipos de transformaciones, como la rotación, traslación, cambio de escala, volteo horizontal, cambio de brillo, cambio de contraste o recorte, por citar algunas.

Cada vez que el modelo reciba una imagen, se aplicarán aleatoriamente las siguientes transformaciones: volteo horizontal, rotación de hasta $\pm 20^\circ$, traslación horizontal y/o vertical de un $\pm 10\%$. En la Figura 5.6 puede verse una comparación entre varias imágenes originales y las imágenes resultantes después del preprocesamiento y el aumento de datos.



Figura 5.6: Imágenes antes y después del preprocesamiento y aumento de datos.

5.4.3. Dropout

Las capas FC suelen ser las causantes del sobreaprendizaje (Srivastava et al., 2014a). Para evitarlo, las capas *dropout* ignoran intermitentemente la señal de salida de las neuronas conectadas a ellas en cada paso del entrenamiento utilizando una probabilidad p configurable. Por ejemplo, si $p = 0,2$, entonces un 20 % de las neuronas serán ignoradas en cada paso del entrenamiento. El efecto del *dropout* puede verse en la Figura 5.7.

En este trabajo se utiliza un *dropout* con probabilidad $p = 0,5$ tras cada capa FC, con excepción de la última.

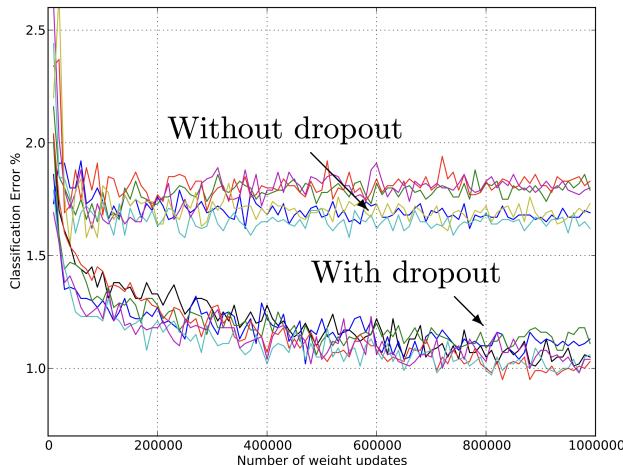


Figura 5.7: Error de varias redes con y sin *dropout* (Srivastava et al., 2014a).

Capítulo 6

Métricas de evaluación

Para evaluar el rendimiento de los diferentes modelos, es necesario definir una serie de métricas. Para explicar todas ellos, se utilizan los siguientes valores:

- VP: Número de verdaderos positivos de una clase; es decir, la cantidad de entradas que pertenecen a una clase c y cuya predicción coincide con la clase c .
- VN: Número de verdaderos negativos de una clase; es decir, la cantidad de entradas que no pertenecen a una clase c y cuya predicción no coincide con la clase c .
- FP: Número de falsos positivos de una clase; es decir, la cantidad de entradas que no pertenecen a una clase c y cuya predicción coincide con la clase c .
- FN: Número de falsos negativos de una clase; es decir, la cantidad de entradas que pertenecen a una clase c y cuya predicción no coincide con la clase c .

6.1. Exactitud

La exactitud de un modelo es la proporción de valores correctamente clasificados. Si, por ejemplo, hay 100 datos a clasificar y 55 de ellos se han clasificado correctamente, la precisión sería de un 55 %.

$$\text{exactitud}_i = \frac{\text{VP}_i + \text{VN}_i}{\text{VP}_i + \text{VN}_i + \text{FP}_i + \text{FN}_i}$$

6.2. Precisión

La precisión es la relación entre los verdaderos positivos de una clase y todas las predicciones cuyo resultado es dicha clase. La precisión global se define como la media de precisión de cada clase. Esta métrica permite ver la calidad de las predicciones positivas.

$$\text{precision} = \frac{\text{VP}_i}{\text{VP}_i + \text{FP}_i}$$

6.3. Recall

El recall se define como la proporción de verdaderos positivos de una clase respecto a la suma de verdaderos positivos y falsos negativos de dicha clase. Esta métrica permite evaluar cuántas instancias positivas fueron identificadas correctamente.

$$\text{recall} = \frac{\text{VP}_i}{\text{VP}_i + \text{FN}_i}$$

6.4. Matriz de confusión

La matriz de confusión permite evaluar visualmente el rendimiento de un modelo. Como refleja su nombre, la matriz de confusión permite ver rápidamente cuánto se equivoca un modelo en sus predicciones, así como las parejas de clases en las que más se equivoca. En la Figura 6.1 se puede observar un ejemplo de matriz de confusión.

	Rabia	Asco	Miedo	Felicidad	Neutral	Tristeza	Sorpresa
Rabia	264	46	19	12	88	43	28
Asco	102	217	19	33	46	61	22
Miedo	19	9	277	12	33	55	95
Felicidad	5	7	1	435	28	4	20
Neutral	34	5	8	35	317	59	42
Tristeza	34	12	19	9	86	317	23
Sorpresa	15	9	63	41	52	32	288

Figura 6.1: Ejemplo de matriz de confusión.

Los elementos de la diagonal de la matriz de confusión representan las predicciones que han acertado. En el eje Y están las clases verdaderas, y en el eje X las clases predichas. Cada celda contiene el número de elementos de la clase correspondiente a la fila que han sido predichos como la categoría correspondiente a la columna. Así, las celdas de la diagonal principal son predicciones correctas, mientras que el resto son incorrectas.

6.5. Métodos de evaluación

Es importante emplear una variedad de métodos para evaluar un modelo y así poder determinar su calidad. Por esto se utilizará la exactitud, la matriz de confusión y la evolución de la función de coste. Se presentarán las métricas de evaluación tanto para el conjunto de entrenamiento como para el de validación, con el fin de realizar una comparación exhaustiva.

Para conseguir evaluar todos los aspectos bajo un contexto profesional y sin sesgos, todos los modelos serán evaluados con el conjunto de datos de validación que AffectNet ofrece para evaluar modelos de FER. Este conjunto cuenta con 500 imágenes de cada clase.

Capítulo 7

Resultados

Todos los modelos alcanzan una exactitud media máxima con el conjunto de prueba mayor al 60 %. Concretamente, VGG10a tiene una exactitud media del 60.43 %, VGG10b tiene una exactitud media del 62.37 %, VGG12 alcanza una exactitud media del 62.54 %, VGG14 consigue una exactitud media del 63.26 %, y VGG17 consigue una exactitud media del 61.37 % (Tabla 7.1). La clase Felicidad es la mejor reconocida en todos los experimentos, con una exactitud que oscila entre el 85.0 % y el 87.0 %. Por otro lado, Asco es la categoría de peores resultados con exactitudes entre 51.2 % y 53.8 %, con la excepción de un 43.4 % con VGG10a. Esta diferencia en resultados probablemente ocurre por la sutileza en varias de las demás expresiones, frente a los rasgos marcados propios de Felicidad (comisuras de los labios hacia arriba, levantamiento de las mejillas, arrugas alrededor de los ojos, entre otras).

Modelo	Rabia	Asco	Miedo	Felicidad	Neutral	Tristeza	Sorpresa	Media
VGG10a	52.8 %	43.4 %	55.4 %	87.0 %	63.4 %	63.4 %	57.6 %	60.43 %
VGG10b	56.2 %	51.2 %	65.4 %	85.2 %	61.2 %	58.2 %	59.2 %	62.37 %
VGG12	59.0 %	53.8 %	59.6 %	85.2 %	57.2 %	61.6 %	61.4 %	62.54 %
VGG14	57.6 %	52.2 %	59.4 %	85.0 %	60.8 %	63.8 %	64.0 %	63.26 %
VGG17	57.0 %	52.0 %	58.6 %	85.8 %	62.6 %	57.6 %	56.0 %	61.37 %

Tabla 7.1: Exactitud de cada modelo por clases con el conjunto de prueba.

Las curvas de exactitud y función de coste en los subconjuntos de entrenamiento y validación son paralelas en todos los modelos, lo que indica que no hay sobreaprendizaje en ninguno de ellos (Figuras 7.1 a 7.10). Curiosamente, el subconjunto de validación proporciona mejores resultados que el subconjunto de entrenamiento. Esto posiblemente se debe a la inactivación del comportamiento de Dropout durante la evaluación.

Las matrices de confusión de todos los modelos muestran que la mayoría de errores de clasificación son comunes (Figuras 7.11 a 7.15). Por ejemplo, todos los modelos confunden notablemente las clases Miedo y Sorpresa, posiblemente porque ambas expresiones comparten varias unidades de acción (Sorpresa es AU1 + AU2 + AU5 + AU26, mientras que Miedo es AU1 + AU2 + AU4 + AU5 + AU7 + AU20 + AU26). Por otro lado, Rabia y Asco también generan confusión en todos

los modelos. Pese a que estas clases no tienen unidades de acción en común, varias de sus imágenes son difíciles de clasificar (por ejemplo, Figura 7.16). Por último, la clase Neutral tiene el mayor número de falsos positivos, ya que varias de sus imágenes contienen expresiones muy sutiles (por ejemplo, Figura 7.17).

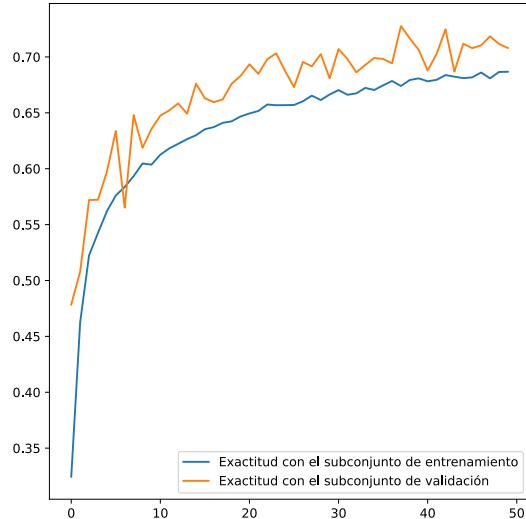


Figura 7.1: Evolución de la exactitud durante el entrenamiento de VGG10a.

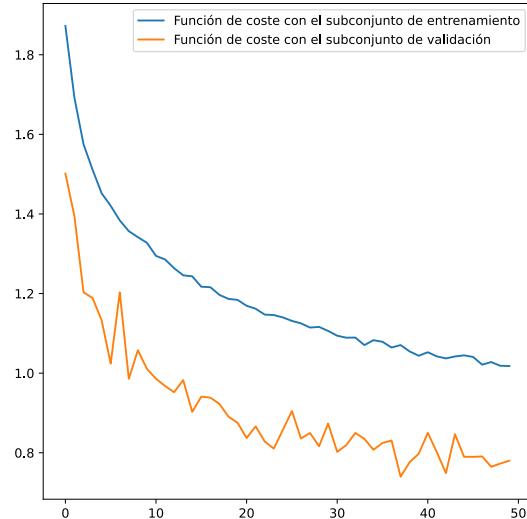


Figura 7.2: Evolución de la función de coste durante el entrenamiento de VGG10a.

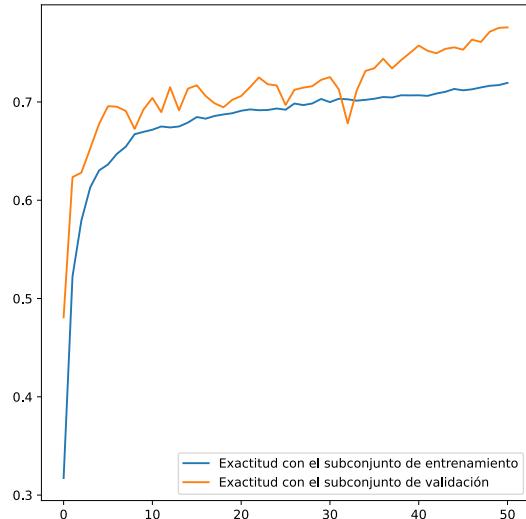


Figura 7.3: Evolución de la exactitud durante el entrenamiento de VGG10b.

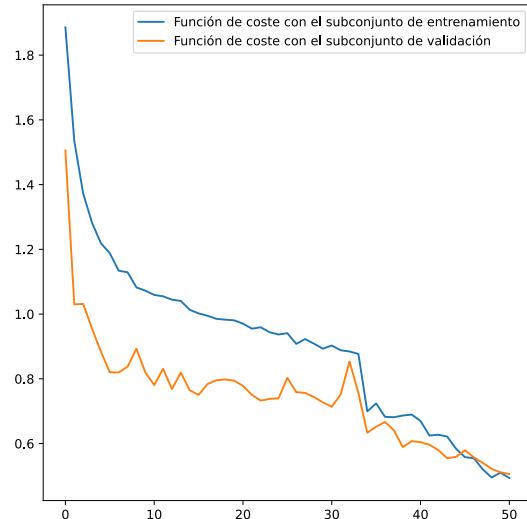


Figura 7.4: Evolución de la función de coste durante el entrenamiento de VGG10b.

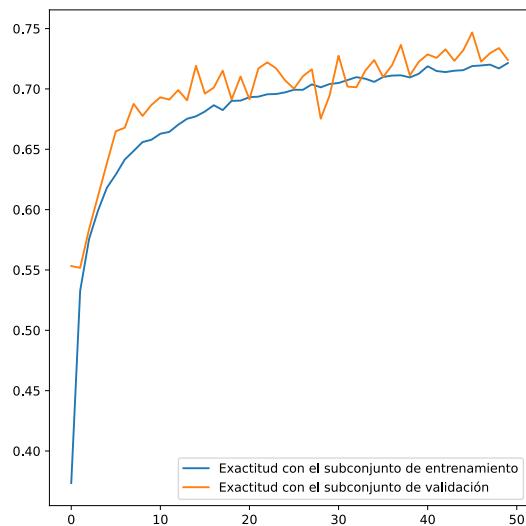


Figura 7.5: Evolución de la exactitud durante el entrenamiento de VGG12.

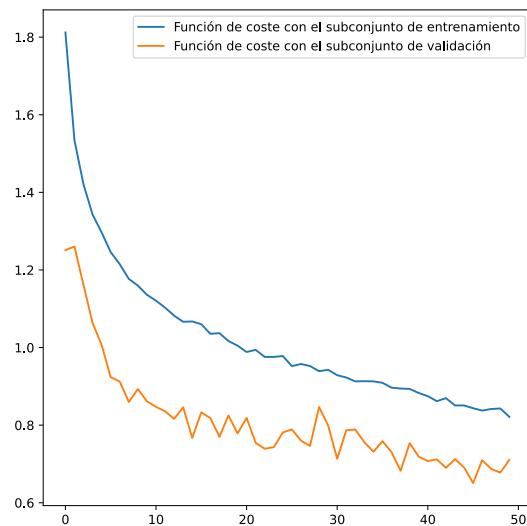


Figura 7.6: Evolución de la función de coste durante el entrenamiento de VGG12.

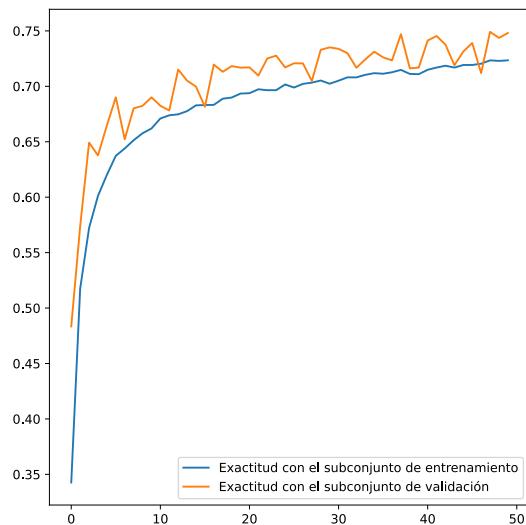


Figura 7.7: Evolución de la exactitud durante el entrenamiento de VGG14.

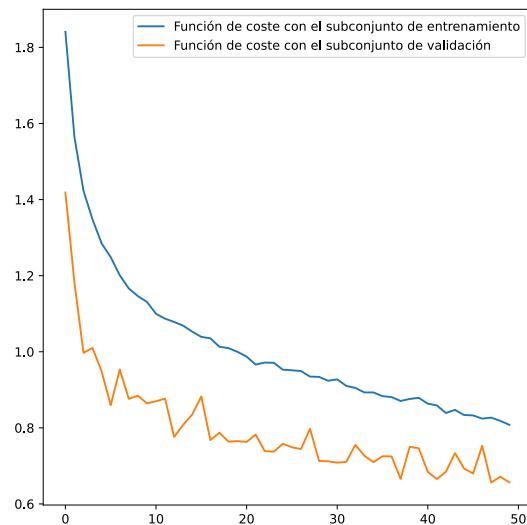


Figura 7.8: Evolución de la función de coste durante el entrenamiento de VGG14.

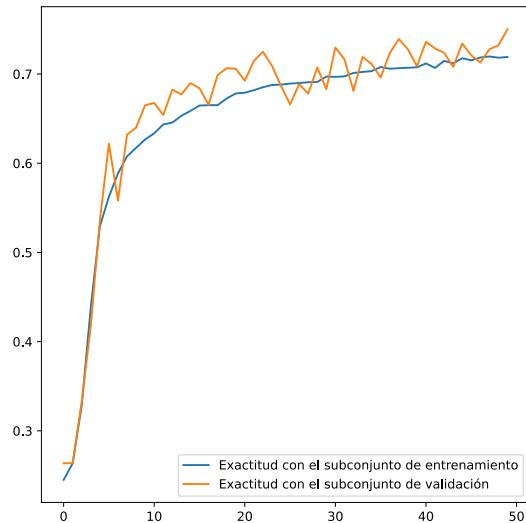


Figura 7.9: Evolución de la exactitud durante el entrenamiento de VGG17.

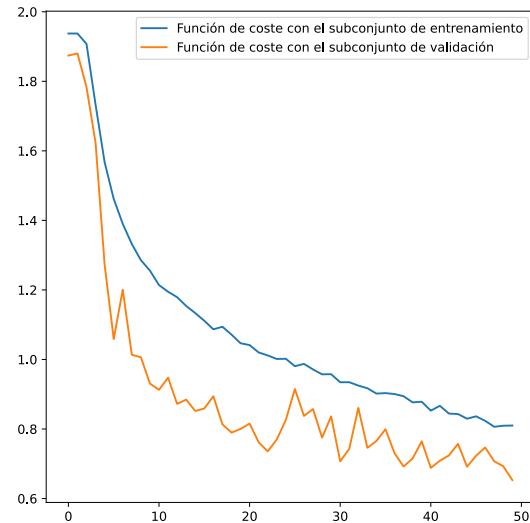


Figura 7.10: Evolución de la función de coste durante el entrenamiento de VGG17.

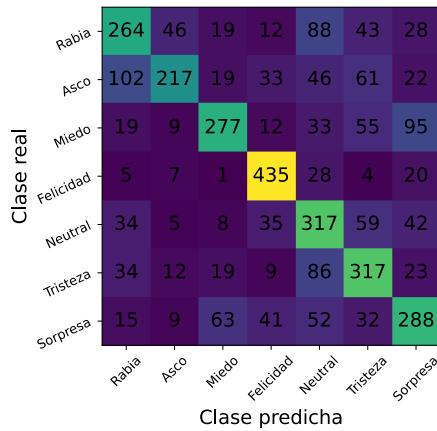


Figura 7.11: Matriz de confusión de VGG10a con el subconjunto de validación público.

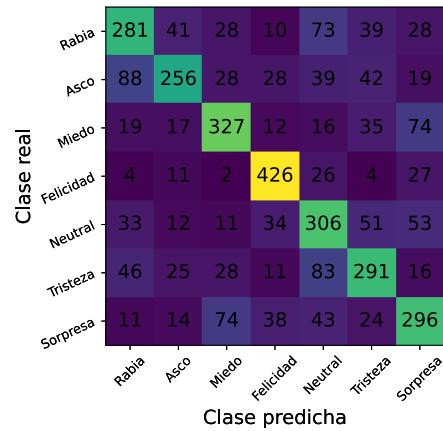


Figura 7.12: Matriz de confusión de VGG10b con el subconjunto de validación público.

	Rabia	Asco	Miedo	Felicidad	Neutral	Tristeza	Sorpresa	
Rabia	295	44	21	12	67	35	26	
Asco	84	269	24	26	29	52	16	
Miedo	20	21	298	11	20	46	84	
Felicidad	4	13	1	426	23	9	24	
Neutral	39	17	8	35	286	61	54	
Tristeza	46	29	18	12	67	308	20	
Sorpresa	13	14	60	33	48	25	307	

Figura 7.13: Matriz de confusión de VGG12 con el subconjunto de validación público.

	Rabia	Asco	Miedo	Felicidad	Neutral	Tristeza	Sorpresa	
Rabia	288	46	23	9	63	39	32	
Asco	89	261	30	27	30	41	22	
Miedo	16	22	297	11	16	44	94	
Felicidad	5	10	1	425	25	6	28	
Neutral	30	12	4	30	304	45	75	
Tristeza	38	24	19	7	66	319	27	
Sorpresa	6	14	60	35	43	22	320	

Figura 7.14: Matriz de confusión de VGG14 con el subconjunto de validación público.

	Rabia	Asco	Miedo	Felicidad	Neutral	Tristeza	Sorpresa	
Rabia	285	52	20	13	74	30	26	
Asco	98	260	23	19	43	42	15	
Miedo	23	14	293	11	28	42	89	
Felicidad	6	10	1	429	25	6	23	
Neutral	41	15	6	28	313	54	43	
Tristeza	52	27	19	11	77	288	26	
Sorpresa	14	17	66	35	67	21	280	

Figura 7.15: Matriz de confusión de VGG17 con el subconjunto de validación público.



Figura 7.16: Confusión entre imágenes de las clases Rabia y Asco.



Figura 7.17: Imágenes de la clase Neutral que pueden interpretarse como otras clases.

Conclusiones y Trabajo Futuro

Conclusiones

El desarrollo de una red neuronal convolucional (CNN) para el reconocimiento de emociones faciales ha sido un desafío que nos ha permitido explorar diversas técnicas y arquitecturas avanzadas. Hemos logrado diseñar, implementar y evaluar un modelo que no solo cumple con los objetivos iniciales, sino que también ofrece un rendimiento competitivo en comparación con otros modelos de vanguardia.

El mejor modelo se basa en la arquitectura VGG14, seleccionada tras múltiples iteraciones y pruebas, debido a su capacidad para equilibrar la complejidad del modelo con su rendimiento. Hemos implementado técnicas avanzadas de preprocesamiento y aumento de datos que han sido cruciales para mejorar la robustez y precisión del modelo.

Además, el uso del optimizador Schedule-free Adam y la función de activación LeakyReLU, ha permitido una convergencia más rápida y estable durante el entrenamiento. Los resultados obtenidos, con una exactitud del 63.26 %, demuestran la efectividad de las CNNs en el reconocimiento de emociones faciales, y la importancia de un preprocesamiento y aumento de datos adecuado para alcanzar este valor.

En conclusión, este trabajo sienta una base sólida para futuras investigaciones en el campo del reconocimiento de emociones faciales, proporcionando un modelo eficiente y una metodología clara para su desarrollo y optimización.

Trabajo Futuro

Las arquitecturas CNN demuestran ser capaces de reconocer emociones faciales en imágenes. No obstante, esta tarea en el mundo real resulta ser más compleja, y este tipo de arquitecturas no son suficientemente potentes para llevarla a cabo. Por esta razón, una posible continuación de este trabajo es experimentar implementando nuevas técnicas con memoria, como LTSM.

Por otro lado, pese a ser el conjunto de datos más grande disponible ahora mismo, AffectNet tiene una distribución de clases muy desequilibrada. El modelo podría verse beneficiado por la introducción de más imágenes de las clases menos representadas, así como la eliminación de imágenes que no se pueden clasificar claramente como Neutral.

Por último, la puesta en marcha de este modelo a través de una interfaz y el estudio de aplicaciones prácticas son también un camino interesante. Una de las

limitaciones del modelo es la necesidad de recortar previamente la sección de una imagen en la que se encuentra el sujeto. La unión de este modelo con un sistema automático de recorte, por ejemplo, podría ser una solución viable.

Conclusions and Future Work

Conclusions

The development of a convolutional neural network (CNN) for facial emotion recognition has been a challenge that has allowed us to explore various advanced techniques and architectures. We have managed to design, implement and evaluate a model that not only meets the initial objectives, but also offers competitive performance compared to other state-of-the-art models.

The best model is based on the VGG14 architecture, selected after multiple iterations and tests due to its ability to balance model complexity with performance. We have implemented advanced preprocessing and data augmentation techniques that have been crucial to improve the robustness and accuracy of the model.

In addition, the use of the Schedule-free Adam optimizer and the LeakyReLU activation function has enabled faster and more stable convergence during training. The results obtained, with an accuracy of 63.26%, demonstrate the effectiveness of CNNs in facial emotion recognition, and the importance of proper data preprocessing and augmentation to achieve this value.

In conclusion, this work lays a solid foundation for future research in the field of facial emotion recognition, providing an efficient model and a clear methodology for its development and optimization

Future Work

CNN architectures can recognize facial emotions in images. However, this task in the real world turns out to be more complex, and such architectures are not powerful enough to perform it. For this reason, a potential follow-up of this work is to try and implement new techniques with memory, such as LSTMs.

On the other hand, despite being the largest dataset available right now, AffectNet has a very unbalanced class distribution. The model could benefit from training with more images of underrepresented classes, as well as removing images that are cannot be clearly classified as Neutral.

Finally, the implementation of this model through an interface and the study of practical applications are also an interesting avenue. One of the limitations of the model is the need to pre-crop the section of the image where the subject is located. Coupling this model with an automated cropping system, for example, could be a viable solution.

Contribuciones Personales

Luis Morales Júlvez

Con el fin de tener un buen control y poder compaginar las clases con el TFG, me encargué de dirigir el grupo. No obstante, cabe destacar que la mayoría de decisiones en este trabajo se tomaron de forma conjunta.

En primer lugar, investigamos sobre cómo se realiza el reconocimiento de expresiones faciales. En mi caso, me encargué de averiguar cuáles fueron los primeros enfoques para llevar a cabo esta tarea. Redacté toda la información que había encontrado y, tras una puesta en común, tomamos la decisión de utilizar CNNs.

A continuación, todos dedicamos un tiempo a aprender más sobre las CNNs y su funcionamiento. Concretamente, yo busqué información sobre la estructura de las redes neuronales y su entrenamiento. Tras compartir toda la información con mis compañeros, preparamos una propuesta de trabajo para el resto del proyecto.

El primer paso de esta propuesta fue determinar la arquitectura de la que participábamos. Tras contemplar varias arquitecturas distintas, elegimos VGG. Cada miembro del grupo propuso varias iteraciones sobre VGG, que luego pusimos en común. De las arquitecturas que acabamos utilizando, yo desarrollé VGG14 y VGG17.

Tras concretar los modelos que pretendíamos entrenar, comenzamos a buscar conjuntos de datos con los que entrenarlos. En mi caso, encontré y propuse el dataset FER-13. Además, escogí las técnicas de preprocesamiento que acabaríamos utilizando.

Para el entrenamiento, hice una implementación de los modelos en TensorFlow. Poco después de empezar a entrenar, nos dimos cuenta de que la capacidad de cómputo de nuestros ordenadores era demasiado pequeña para completar este proyecto. Para solventar este problema, cada miembro investigó un servicio de computación en la nube distinto. En mi caso, busqué información sobre la plataforma AWS. Finalmente, decidimos utilizar *Google Cloud* porque nos proporcionaba más créditos que otras plataformas y nos permitía utilizar GPUs.

Una vez entrenados todos los modelos, ayudé a Francisco a preparar los *scripts* para generar las gráficas a partir de las métricas que habíamos extraído. Junto a mis compañeros, estudié los resultados obtenidos y colaboré en la redacción de conclusiones.

Finalmente, ayudé a mis compañeros a redactar los capítulos restantes de la memoria. Concretamente, me encargué de la redacción del estado del arte, arquitectura, conclusiones y, junto a Jaime, el capítulo de experimentación.

Jaime Costas Insua

En primer lugar, investigamos sobre cómo se realiza el reconocimiento de expresiones faciales. En mi caso, me encargué de averiguar para qué servían y cómo funcionaban las diferentes arquitecturas mixtas. Redacté toda la información que había encontrado y, tras una puesta en común, tomamos la decisión de utilizar CNNs.

A continuación, todos dedicamos un tiempo a aprender más sobre las CNNs y su funcionamiento. Concretamente, yo busqué información sobre cómo implementar y entrenar una CNN. Tras compartir toda la información con mis compañeros, preparamos una propuesta de trabajo para el resto del proyecto.

El primer paso de esta propuesta fue determinar la arquitectura de la que participaríamos. Tras contemplar varias arquitecturas distintas, elegimos VGG. Cada miembro del grupo propuso varias iteraciones sobre VGG, que luego pusimos en común. De las arquitecturas que acabamos utilizando, yo desarrollé VGG10a y VGG10b.

Tras concretar los modelos que pretendíamos entrenar, comenzamos a buscar conjuntos de datos con los que entrenarlos. En mi caso, encontré y propuse el dataset FER-13. Además, escogí las técnicas de preprocessamiento que acabaríamos utilizando.

Para el entrenamiento, hice una implementación de los modelos en Pytorch. Poco después de empezar a entrenar, nos dimos cuenta de que la capacidad de cómputo de nuestros ordenadores era demasiado pequeña para completar este proyecto. Para solventar este problema, cada miembro investigó un servicio de computación en la nube distinto. En mi caso, busqué información sobre la plataforma Microsoft Azure. Finalmente, decidimos utilizar *Google Cloud* porque nos proporcionaba más créditos que otras plataformas y nos permitía utilizar GPUs.

Preparé el código necesario para preprocessar los datos y entrenar los modelos y, tras el entrenamiento, estudié los resultados obtenidos junto a mis compañeros y colaboré en la redacción de conclusiones.

En lo referido a la memoria, me encargué de organizar los diferentes capítulos, de asegurarme de que el estilo de redacción fuese consistente a lo largo de todo el proyecto y de la revisión final. También redacté el capítulo de la introducción, el resumen, las métricas de evaluación y, junto a Luis, el capítulo de experimentación.

Francisco Calvo González

En primer lugar, investigamos sobre cómo se realiza el reconocimiento de expresiones faciales. Concretamente, me encargué de averiguar qué resultados se podían esperar de un modelo CNN. Redacté toda la información que había encontrado y, tras una puesta en común, tomamos la decisión de utilizar CNNs.

A continuación, todos dedicamos un tiempo a aprender más sobre las CNNs y su funcionamiento. Concretamente, yo busqué información sobre las capas convolucionales, los filtros y las arquitecturas típicas. Tras compartir toda la información con mis compañeros, preparamos una propuesta de trabajo para el resto del proyecto.

El primer paso de esta propuesta fue determinar la arquitectura de la que parti-

ríamos. Tras contemplar varias arquitecturas distintas, elegimos VGG. Cada miembro del grupo propuso varias iteraciones sobre VGG, que luego pusimos en común. De las arquitecturas que acabamos utilizando, yo desarrollé VGG12.

Tras concretar los modelos que pretendíamos entrenar, comenzamos a buscar conjuntos de datos con los que entrenarlos. En mi caso, encontré y propuse el dataset RAF-DB. Además, escogí las técnicas de aumento de datos que acabaríamos utilizando.

Poco después de empezar a entrenar, nos dimos cuenta de que la capacidad de cómputo de nuestros ordenadores era demasiado pequeña para completar este proyecto. Para solventar este problema, cada miembro investigó un servicio de computación en la nube distinto. En mi caso, busqué información sobre la plataforma *Google Cloud*. Hice varias pruebas con TPUs y GPUs hasta encontrar la configuración que acabaríamos utilizando.

Una vez entrenados todos los modelos, preparé los *scripts* para generar las gráficas a partir de las métricas que habíamos extraído. Junto a mis compañeros, estudié los resultados obtenidos y colaboré en la redacción de conclusiones.

Finalmente, ayudé a mis compañeros a redactar los capítulos restantes de la memoria. Concretamente, me encargué de la redacción de los conceptos teóricos de las redes neuronales y la sección de resultados, así como de la elaboración de la mayoría de gráficos e imágenes.

Bibliografía

- AKHTAR, M. J., MAHUM, R., BUTT, F. S., AMIN, R., EL-SHERBENY, A. M., LEE, S. M. y SHAIKH, S. A robust framework for object detection in a traffic surveillance system. *Electronics*, vol. 11(21), página 3425, 2022.
- BROWNLEE, J. A gentle introduction to cross-entropy for machine learning. 2020.
- CHEN, Y., WANG, J., CHEN, S., SHI, Z. y CAI, J. Facial motion prior networks for facial expression recognition. 2019.
- DEFAZIO, A., YANG, X. A., MISHCHENKO, K., CUTKOSKY, A., MEHTA, H. y KHALED, A. Schedule-free optimization in pytorch. 2024.
- EKMAN, P. y FRIESEN, W. V. Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology*, vol. 17(2), página 124–129, 1971.
- EKMAN, P. y FRIESEN, W. V. Facial action coding system. 1978.
- FASEL, B. Head-pose invariant facial expression recognition using convolutional neural networks. En *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*, páginas 529–534. 2002.
- FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, vol. 36(4), página 193–202, 1980.
- GEORGESCU, M.-I., IONESCU, R. T. y POPESCU, M. Local learning with deep and handcrafted features for facial expression recognition. *IEEE Access*, vol. 7, página 64827–64836, 2019.
- GOODFELLOW, I. J., ERHAN, D., CARRIER, P. L., COURVILLE, A., MIRZA, M., HAMNER, B., CUKIERSKI, W., TANG, Y., THALER, D., LEE, D.-H., ZHOU, Y., RAMAIAH, C., FENG, F., LI, R., WANG, X., ATHANASAKIS, D., SHAWETAYLOR, J., MILAKOV, M., PARK, J., IONESCU, R., POPESCU, M., GROZEA, C., BERGSTRA, J., XIE, J., ROMASZKO, L., XU, B., CHUANG, Z. y BENGIO, Y. Challenges in representation learning: A report on three machine learning contests. 2013.
- HE, K., ZHANG, X., REN, S. y SUN, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- IDE, H. y KURITA, T. Improvement of learning for cnn with relu activation by sparse regularization. *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.
- KINGMA, D. P. y BA, J. Adam: A method for stochastic optimization. 2017.
- KOUSHIK. Understanding convolutional neural networks (cnns) in depth. 2023.
- KRIZHEVSKY, A., SUTSKEVER, I. y HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, vol. 60(6), página 84–90, 2017.
- LI, S. y DENG, W. Reliable crowdsourcing and deep locality-preserving learning for unconstrained facial expression recognition. *IEEE Transactions on Image Processing*, vol. 28(1), páginas 356–370, 2019.
- LI, Y., ZENG, J., SHAN, S. y CHEN, X. Occlusion aware facial expression recognition using cnn with attention mechanism. *IEEE Transactions on Image Processing*, vol. 28(5), páginas 2439–2450, 2019.
- LIEN, J., KANADE, T., COHN, J. y LI, C.-C. Automated facial expression recognition based on faces action units. *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998.
- LUCEY, P., COHN, J. F., KANADE, T., SARAGIH, J., AMBADAR, Z. y MATTHEWS, I. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. En *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, páginas 94–101. 2010.
- JU MAO, J., XU, R., YIN, X., CHANG, Y., NIE, B. y HUANG, A. Poster v2: A simpler and stronger facial expression recognition network. *ArXiv*, vol. abs/2301.12149, 2023.
- MATSUGU, M., MORI, K., MITARI, Y. y KANEDA, Y. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, vol. 16(5), páginas 555–559, 2003. ISSN 0893-6080. Advances in Neural Networks Research: IJCNN '03.
- MCCULLOCH, W. S. y PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, vol. 5(4), página 115–133, 1943.
- MINSKY, M. y PAPERT, S. *Perceptrons; an Introduction to Computational Geometry*. MIT Press, 1969. ISBN 9780262630221.
- MOLLAHOSSEINI, A., HASANI, B. y MAHOOR, M. H. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, vol. 10(1), página 18–31, 2019.
- NABIL, M. Unveiling the diversity: A comprehensive guide to types of cnn architectures. 2023.

- NELSON, C. A. The recognition of facial expressions in the first two years of life: Mechanisms of development. *Child Development*, vol. 58(4), páginas 889–909, 1987. ISSN 00093920, 14678624.
- NIE, W., YAN, Y., SONG, D. y WANG, K. Multi-modal feature fusion based on multi-layers lstm for video emotion recognition. *Multimedia Tools and Applications*, vol. 80(11), página 16205–16214, 2020.
- OOMMEN, T., MISRA, D., TWARAKAVI, N. K., PRAKASH, A., SAHOO, B. y BANDOPADHYAY, S. An objective analysis of support vector machine based classification for remote sensing. *Mathematical Geosciences*, vol. 40(4), página 409–424, 2008.
- PITALOKA, D. A., WULANDARI, A., BASARUDDIN, T. y LILIANA, D. Y. Enhancing cnn with preprocessing stage in automatic emotion recognition. *Procedia Computer Science*, vol. 116, página 523–529, 2017.
- RICHTER, M. L., BYTTNER, W., KRUMNACK, U., WIEDENROTH, A., SCHALLNER, L. y SHENK, J. (input) size matters for cnn classifiers. *Lecture Notes in Computer Science*, página 133–144, 2021.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, vol. 65(6), página 386–408, 1958.
- SEPÚLVEDA, J. A. C. La importancia de la optimización en el aprendizaje automático (machine learning). 2023.
- SIMONYAN, K. y ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. 2015.
- SINGH, L. Deep bi-directional lstm network with cnn features for human emotion recognition in audio-video signals. *International Journal of Swarm Intelligence*, vol. 1(1), página 1, 2022.
- SOTO, C. Entendiendo cnn (convolutional neural network) redes neuronales convolucionales. 2022.
- SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I. y SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, vol. 15(56), páginas 1929–1958, 2014a.
- SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I. y SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, vol. 15(1), páginas 1929–1958, 2014b.
- SUN, B., WEI, Q., LI, L., XU, Q., HE, J. y YU, L. Lstm for dynamic emotion and group emotion recognition in the wild. *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, 2016.

- SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V. y RABINOVICH, A. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- VADAPALLI, H. Recognition of facial action units from video streams with recurrent neural networks : a new paradigm for facial expression recognition. 2011.
- VAISHNAV, R. Handwritten digit recognition using pytorch. 2020.
- VERSLOOT, C. Leaky relu: Improving traditional relu. 2019.
- WANG, Y., WU, J. y HOASHI, K. Lightweight deep convolutional neural networks for facial expression recognition. En *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, páginas 1–6. 2019.
- YUDIN, D., , y KAPUSTINA, E. *The Usage of Grayscale or Color Images for Facial Expression Recognition with Deep Neural Networks*, páginas 271–281. 2020. ISBN 978-3-030-30424-9.
- ZENG, J., SHAN, S. y CHEN, X. Facial expression recognition with inconsistently annotated datasets. En *Computer Vision – ECCV 2018* (editado por V. Ferrari, M. Hebert, C. Sminchisescu y Y. Weiss), páginas 227–243. Springer International Publishing, Cham, 2018. ISBN 978-3-030-01261-8.
- ZHANG, S., ZHANG, Y., ZHANG, Y., WANG, Y. y SONG, Z. A dual-direction attention mixed feature network for facial expression recognition. *Electronics*, vol. 12(17), página 3595, 2023.
- ZHENG, C., MENDIETA, M. y CHEN, C. Poster: A pyramid cross-fusion transformer network for facial expression recognition. *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2023.