Desarrollo y Evaluación de Modelos de Análisis de Sentimientos en Imágenes y Textos con IA Explicable y Propuesta de Integración para Estudios de Mercado

Diego Alejandro Cordova Barrera

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo y Evaluación de Modelos de Análisis de Sentimientos en Imágenes y Textos con IA Explicable y Propuesta de Integración para Estudios de Mercado

Trabajo de graduación en modalidad de Trabajo Profesional presentado por

Diego Alejandro Cordova Barrera

Para optar al grado académico de Licenciado en Ciencias de la Computación y TI

Guatemala, octubre del 2024

Vo.Bo.:

(f) _____

MSc, MBA, Ing. Luis Alberto Suriano Saravia

Tribunal Examinador:

(f) _____

MSc, MBA, Ing. Luis Alberto Suriano Saravia

(f) _____

(f) _____

Fecha de aprobación: Guatemala, .

Prefacio

Quiero expresar mi más sincero agradecimiento a todas las personas y entidades que hicieron posible la realización de este trabajo de graduación. En primer lugar agradezco a mi familia, por su amor y apoyo incondicional. Gracias por comprender mis ausencias y por ser siempre mi mayor fuente de inspiración y fortaleza. A mis padres Dario Cordova e Imelda Barrera, por enseñarme la importancia del esfuerzo y la perseverancia, y por brindarme cada oportunidad para perseguir mis sueños. A mis amigos y compañeros, quienes con sus palabras de ánimo y compañía hicieron que este viaje fuera más llevadero.

A mi asesor, Alberto Suriano, por su invaluable guía, paciencia y dedicación a lo largo de este proceso. Su conocimiento y sus recomendaciones fueron fundamentales para el desarrollo de este trabajo, y su apoyo constante me motivó a superar cada obstáculo.

Agradezco también a la Universidad del Valle de Guatemala y a sus profesores, quienes compartieron conmigo sus conocimientos y experiencias, formando una base sólida para el desarrollo de mi carrera. Gracias a todos los compañeros que colaboraron en este proyecto, por sus aportes y sus valiosas opiniones.

Índice

Lista de Figuras

Lista de Cuadros

Resumen

Este proyecto propone el desarrollo de un sistema integral de análisis de emociones para estudios de mercado, que combina técnicas de visión por computadora y procesamiento de lenguaje natural. En primer lugar, se implementa un detector de rostros basado en filtros de Haar y detectores en cascada, capaz de identificar rostros en tiempo real con alta precisión. A partir de los rostros detectados, se entrena un modelo de redes neuronales convolucionales (CNN) para la clasificación de expresiones faciales, lo cual permite captar las reacciones emocionales de forma automática. Complementariamente, se diseña un modelo de análisis de sentimiento en texto, enfocado en interpretar y clasificar las emociones presentes en comentarios textuales.

Para garantizar la transparencia y confiabilidad de ambos modelos, se aplican técnicas de Explainable AI, permitiendo una comprensión profunda de las características clave que influyen en sus decisiones.

Finalmente, se propone un *pipeline* que integra ambos modelos, el cual se presenta como una herramienta innovadora para estudios de mercado, ya que posibilita el análisis simultáneo de expresiones faciales y comentarios textuales, ofreciendo una perspectiva más completa sobre las percepciones y emociones de los usuarios. Este enfoque busca mejorar la precisión y relevancia de los análisis en el ámbito del mercadeo, contribuyendo al diseño de estrategias más efectivas y alineadas con las necesidades de los consumidores.

Abstract

This project proposes the development of a comprehensive emotion analysis system for market research, combining computer vision and natural language processing techniques. First, a face detector based on Haar filters and cascade classifiers is implemented, capable of identifying faces in real time with high accuracy. Using the detected faces, a Convolutional Neural Network (CNN) model is trained for facial expression classification, allowing for automatic capture of emotional reactions. Additionally, a sentiment analysis model is designed to interpret and classify emotions present in textual comments.

To ensure the transparency and reliability of both models, Explainable AI techniques are applied, providing an in-depth understanding of the key features that influence their decisions.

Finally, a pipeline that integrates both models is proposed, presenting itself as an innovative tool for market research by enabling simultaneous analysis of facial expressions and textual comments, thus offering a more comprehensive perspective on user perceptions and emotions. This approach seeks to enhance the accuracy and relevance of analyses within the marketing field, contributing to the design of more effective strategies that align with consumer needs.

CAPÍTULO 1

Introducción

En los últimos años, la inteligencia artificial (IA) ha emergido como una poderosa herramienta en varios ámbitos del marketing, demostrando ser extremadamente efectiva. Actualmente, el 75 % de los profesionales de mercadeo tienen acceso a herramientas de inteligencia artificial y el 60 % de estos profesionales indican que ha ayudado a crear experiencias centradas en el usuario [10]. El AI Marketing, conocido también como Marketing de Inteligencia Artificial, aprovecha el poder de la IA para automatizar los procesos de toma de decisiones en marketing. Esto implica la utilización de tecnologías de IA para recopilar y analizar datos, así como para observar el comportamiento de la audiencia y las tendencias económicas que influyen en las iniciativas de marketing. En particular, en el marketing digital, donde la rapidez es crucial, la IA se emplea para agilizar los esfuerzos [3]. En este contexto, el presente trabajo se enfoca en el desarrollo de modelos de análisis de sentimientos tanto en rostros como en texto, con el objetivo de proponer una integración de ambos para aplicaciones en estudios de mercado. Cabe mencionar que la incorporación de IA Explicable (XAI) será un componente esencial en ambos modelos, permitiendo entender y justificar las decisiones tomadas por los algoritmos, lo cual es fundamental en entornos de marketing donde la transparencia y la interpretabilidad son claves. Este trabajo se estructura de la siguiente manera: inicialmente, se presenta un marco teórico que cubre los conceptos básicos y la relevancia del análisis de sentimientos y XAI. Luego, se describen detalladamente los métodos y datos utilizados para desarrollar los modelos de análisis de sentimientos en rostros y texto. Posteriormente, se propone una arquitectura para integrar ambos modelos y se discuten sus posibles aplicaciones en estudios de mercado. Finalmente, se presentan los resultados y se concluye con una discusión sobre el impacto potencial y las direcciones futuras para la investigación.

CAPÍTULO 2

Objetivos

2.1. Objetivo General

Desarrollar un modelo de detección de sentimientos en rostros en tiempo real y un modelo de análisis de sentimientos en texto, incorporando técnicas de Explainable AI para ambos. Este proyecto servirá como base para crear un pipeline que pueda ser aplicado en estrategias de marketing, aunque dicha aplicación no es el objetivo principal del proyecto.

2.2. Objetivos Específicos

- Desarrollar un algoritmo de detección de rostros utilizando filtros de Haar y método cascada el cual recorte los rostros encontrados en un video de tiempo real.
- Desarrollar un modelo de análisis de sentimientos faciales capaz de clasificar las expresiones emocionales en tiempo real a partir de los rostros encontrados por el algoritmo de detección.
- Desarrollar un modelo de análisis de sentimientos en texto que sea capaz de clasificar la emoción encontrada en texto.
- Explicar el funcionamiento y métricas de los 2 modelos utilizando técnicas de Explainable AI para ambos.
- Elaborar una propuesta de pipeline que utilice estos modelos en conjunto y proponer posibles aplicaciones prácticas en el ámbito del mercadeo.

CAPÍTULO 3

Justificación

En la década de 2010 y 2020, con el crecimiento exponencial de los datos (big data), los algoritmos de machine learning se volvieron esenciales. Varias empresas utilizaron estos algoritmos para crear motores de recomendación sofisticados. Por otro lado, el deep learning, una rama más sofisticada del aprendizaje automático, permitió el marketing hiper personalizado. Esto se logró desarrollando herramientas de análisis de sentimiento para analizar grandes cantidades de datos de redes sociales, blogs y foros en tiempo real, permitiendo a los profesionales de marketing evaluar el sentimiento público hacia sus marcas [12]. A pesar de los avances notables, la aplicación de modelos de inteligencia artificial para el marketing en tiempo real enfrenta desafíos significativos. Entre estos desafíos se encuentran los costos económicos asociados con la implementación de estas herramientas, la preocupación por la seguridad y transparencia de los datos utilizados, así como la falta de conocimiento en los campos de inteligencia artificial y aprendizaje profundo necesarios para implementar soluciones efectivas [2]. Por lo tanto, el presente proyecto propone llenar esta brecha en la investigación mediante el desarrollo de modelos de análisis de sentimientos tanto en rostros como en texto, con el objetivo de proponer una integración de ambos para aplicaciones en estudios de mercado.Al abordar esta problemática, se espera contribuir al avance del conocimiento en el campo del análisis de sentimientos faciales y textuales, así como su aplicación en el ámbito del marketing y el diseño de productos. Además, este proyecto tiene el potencial de ofrecer beneficios prácticos significativos, tanto para los diseñadores de productos como para los profesionales del marketing. Al proporcionar una comprensión más profunda de cómo las emociones de los consumidores se relacionan con las características del producto y cómo son procesadas por los algoritmos de inteligencia artificial.

CAPÍTULO 4

Marco Teórico

4.1. Análisis de Sentimiento

El análisis de sentimiento es un proceso en el campo del procesamiento de lenguaje natural (NLP) que se centra en identificar y clasificar opiniones o emociones expresadas en textos. Este análisis permite discernir si el sentimiento predominante en un texto es positivo, negativo o neutral, proporcionando una comprensión más profunda de la percepción y las emociones del público sobre ciertos temas. En el contexto de las redes sociales, como Twitter, el análisis de sentimiento es especialmente útil para examinar grandes volúmenes de datos generados por los usuarios, permitiendo conocer tendencias de opinión, dinámicas de mercado y reacciones ante marcas o eventos. Para realizar este análisis, se emplean enfoques que incluyen técnicas de aprendizaje automático y vectores de características, logrando clasificar textos en función de su polaridad emocional con modelos que pueden ser desde máquinas de vectores de soporte hasta redes neuronales recurrentes o modelos avanzados como BERT [18].

4.2. Detectores de Cascada

Un detector en cascada consiste en una serie de etapas con estructura jerárquica y secuencial donde cada etapa del detector actúa como un nivel en una cascada . Cada etapa está diseñada para clasificar una subventana de la imagen como .ºbjeto.º "no objeto". Si una subventana es clasificada como "no objeto.en cualquier etapa, se descarta inmediatamente, y no se evalúa en las etapas posteriores. Esto permite que el sistema sea muy eficiente, ya que se evita el procesamiento innecesario de subventanas que probablemente no contienen el objeto de interés [21].

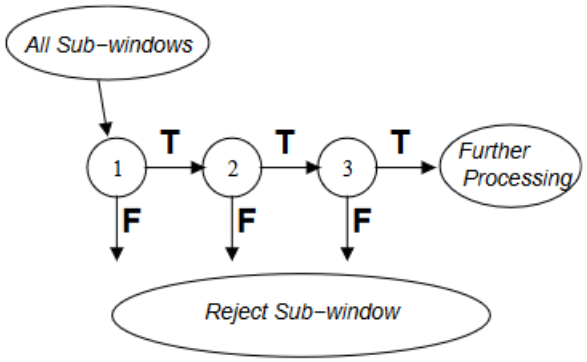

Figura 1: Representación de la cascada de detección.

[21]

4.2.1. Filtros de Haar

Estos filtros son utilizados para extraer características de una imagen basándose en la diferencia de intensidades entre áreas adyacentes para representar bordes y texturas visuales. Estos filtros funcionan de manera similar a las funciones escalonadas de base de Haar en matemáticas [21].

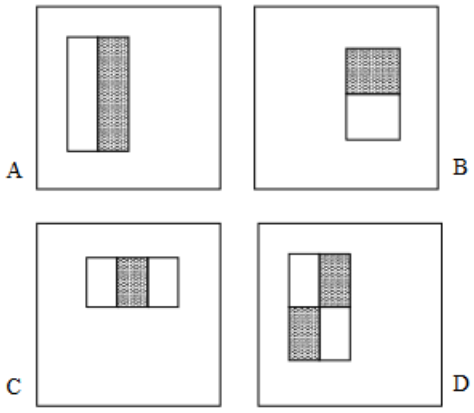

Figura 2: Ejemplo de filtros de Haar utilizados para la detección de características en imágenes.

[21]

Los filtros también consisten en rectángulos adyacentes que comparan la intensidad de las regiones, permitiendo detectar con precisión bordes y texturas clave en la identificación de objetos [21].

4.2.2. Imágen Integral

La imagen integral es una representación que permite calcular de manera eficiente la suma de los valores de los píxeles dentro de un rectángulo definido en la imagen original. Para cada píxel en la imagen integral, su valor corresponde a la suma de todos los píxeles que se encuentran por encima y a la izquierda de ese punto, incluyendo el propio. Una vez creada la imagen integral, se pueden calcular características como los filtros de Haar en cualquier escala y ubicación de forma constante. Esto es posible porque, en lugar de sumar manualmente los valores de los píxeles cada vez que se evalúa una característica, el resultado puede obtenerse con solo cuatro accesos a la imagen integral, lo que reduce significativamente el tiempo de procesamiento. La imagen integral en la ubicación x,yx,y contiene la suma de los píxeles situados por encima y a la izquierda de x,yx,y inclusivo. Como se muestra en la siguiente ecuación [21].

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

La imagen integral permite que el sistema de detección de objetos funcione de manera más rápida y eficiente, lo que es crucial para aplicaciones en tiempo real, como la detección de rostros. Al poder calcular rápidamente las características necesarias, el sistema puede evaluar muchas más ubicaciones en la imagen en un tiempo dado, mejorando así la tasa de detección y reduciendo la cantidad de falsos negativo [21].

4.2.3. Algoritmo AdaBoost

AdaBoost es un algoritmo de aprendizaje automático que mejora la precisión de los clasificadores combinando múltiples clasificadores débiles para formar uno fuerte. En cada iteración, entrena un nuevo clasificador y ajusta los pesos de las instancias de entrenamiento, dando mayor importancia a los errores previos. Los clasificadores se combinan según su precisión, con los más precisos influyendo más en la clasificación final. Aunque es efectivo y fácil de implementar, puede ser sensible a los ruidos y outliers. AdaBoost se utiliza ampliamente en tareas como la detección de objetos y el reconocimiento de patrones [21].

A continuación se describe el algoritmo:

- Dados ejemplos de imágenes $(x_1, y_1), \ldots, (x_n, y_n)$, donde $y_i = 0, 1$ para ejemplos negativos y positivos respectivamente.
- Inicializar los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0, 1$ respectivamente, donde m y l son el número de negativos y positivos respectivamente.
- Para $t = 1, \ldots, T$:

1. Normalizar los pesos:
$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
para que w_t sea una distribución de probabilidad.
2. Para cada característica j, entrenar un clasificador h_j que se restrinja a usar una única característica. El error se evalúa respecto a w_t:
$$\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|.$$
3. Escoger el clasificador h_t con el menor error ϵ_t.
4. Actualizar los pesos:
$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
donde $e_i = 0$ si el ejemplo x_i se clasifica correctamente, $e_i = 1$ en caso contrario, y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- El clasificador fuerte final es:
$$h(x) = \begin{cases} 1 & \text{si } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \\ 0 & \text{en otro caso.} \end{cases}$$
donde $\alpha_t = \log \frac{1}{\beta_t}$.

4.3. Machine Learning

El aprendizaje automático (ML, por sus siglas en inglés) es un subconjunto de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos estadísticos que permiten a las computadoras realizar tareas específicas sin una programación explícita [1]. Involucra el uso de datos para entrenar modelos, lo que les permite reconocer patrones, tomar decisiones y mejorar su desempeño con el tiempo, basado en la experiencia [1].

En mayor detalle, el aprendizaje automático puede caracterizarse por los siguientes componentes clave:

1. **Aprendizaje Basado en Datos:** El ML se basa en grandes conjuntos de datos para entrenar modelos. La calidad y cantidad de los datos influyen significativamente en la capacidad del modelo para aprender y generalizar a partir del proceso de entrenamiento [1].
2. **Tipos de Aprendizaje:** El aprendizaje automático incluye varios paradigmas, entre ellos:
   - **Aprendizaje Supervisado:** El modelo se entrena con datos etiquetados, donde se proporciona la salida correcta, lo que le permite aprender la relación entre entradas y salidas [1].

- **Aprendizaje No Supervisado:** El modelo trabaja con datos sin etiquetar, identificando patrones y estructuras sin salidas predefinidas [1].
- **Aprendizaje por Refuerzo:** El modelo aprende interactuando con un entorno, recibiendo retroalimentación en forma de recompensas o penalizaciones en función de sus acciones [1].

3. **Mejora del Desempeño:** El objetivo principal del aprendizaje automático es mejorar el desempeño en una tarea específica (T), medida mediante una métrica de rendimiento (P). Esta mejora se logra mediante un entrenamiento iterativo, en el que el modelo ajusta sus parámetros en función de los datos procesados [1].
4. **Adaptabilidad:** Los modelos de ML están diseñados para adaptarse a nuevos datos y entornos cambiantes, lo que los hace adecuados para aplicaciones dinámicas, como sistemas de recomendación, detección de fraudes y sistemas autónomos [1].
5. **Aplicaciones:** El aprendizaje automático tiene una amplia gama de aplicaciones en diversos dominios, incluidos la salud (diagnóstico médico), finanzas (puntuación de crédito), marketing (segmentación de clientes) y tecnología (procesamiento de lenguaje natural, reconocimiento de imágenes) [1].

En general, el aprendizaje automático representa un enfoque poderoso para la resolución de problemas que aprovecha los datos para permitir que los sistemas aprendan, se adapten y tomen decisiones informadas, mejorando su capacidad para realizar tareas complejas con una mínima intervención humana [1].

4.4. Aprendizaje profundo

El aprendizaje profundo o *deep learning* es una subcategoría del aprendizaje automático que involucra modelos computacionales compuestos por múltiples capas de procesamiento, lo que permite que estos modelos aprendan representaciones de datos con varios niveles de abstracción. Utiliza arquitecturas como las redes neuronales profundas, que pueden descubrir automáticamente estructuras complejas en grandes conjuntos de datos [14].

4.5. Redes Neuronales

Las redes neuronales son modelos computacionales inspirados en las redes neuronales biológicas que consisten en capas interconectadas de nodos o "neuronas", donde cada conexión tiene un peso asociado. Las redes neuronales procesan los datos de entrada pasándolos a través de múltiples capas, cada una de las cuales realiza transformaciones a través de sumas ponderadas seguidas de funciones de activación no lineales. Esta arquitectura permite que las redes neuronales aprendan patrones

y representaciones complejos a partir de datos, lo que las hace efectivas para tareas como clasificación, regresión y extracción de características. El entrenamiento de redes neuronales generalmente implica ajustar los pesos utilizando algoritmos como la retropropagación, que optimiza el rendimiento de la red en una tarea determinada al minimizar el error entre las salidas previstas y reales [14].

Las redes neuronales se organizan en capas, cada una compuesta por un conjunto de neuronas que se conectan con las neuronas de las capas adyacentes mediante conexiones ponderadas. Estas capas se pueden clasificar en tres tipos principales: la capa de entrada, donde las neuronas reciben la información inicial que se procesará; las capas ocultas, que suelen ser más numerosas que la capa de entrada, ya que son responsables de identificar patrones y extraer características más abstractas a partir de los datos de entrada; y, finalmente, la capa de salida, que proporciona los resultados procesados, como predicciones o clasificaciones, dependiendo del problema que esté resolviendo la red. En cada neurona, las entradas se transforman mediante funciones de activación, que introducen no linealidades cruciales para que la red neuronal sea capaz de modelar relaciones complejas. Entre las funciones de activación más comunes se encuentran la función sigmoide, la tangente hiperbólica (tanh) y la función rectificada lineal (ReLU). La cantidad de capas y neuronas puede variar según la complejidad del problema, y es común que las redes profundas incluyan muchas capas ocultas para lograr un mayor nivel de abstracción en el análisis de los datos [14].

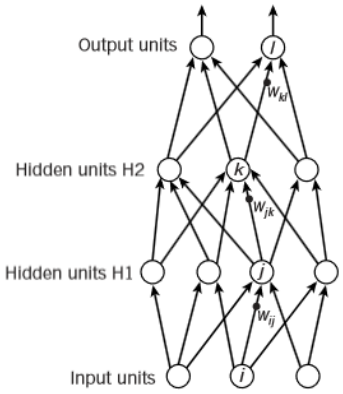

Figura 3: Figura 3: Diagrama de una red neuronal.

[14]

4.5.1. Retropropagación

Las redes neuronales ajustan sus pesos o ponderaciones entre las conexiones de neuronas a través de un proceso que se conoce como retropropagación. El objetivo principal de la retropropagación es minimizar el error entre la salida predicha por la red y la salida real esperada, mejorando la precisión de las predicciones de la red a lo largo del tiempo. Este ajuste se realiza mediante la transmisión hacia atrás

(back propagation) del error a través de la red, desde la capa de salida hasta las capas ocultas e, incluso, hasta la capa de entrada. A través de un cálculo basado en derivadas parciales, la retropropagación mide cómo afecta cada peso de la red al error total. Utilizando técnicas de optimización como el descenso por gradiente, la red actualiza sus pesos para reducir la magnitud del error de manera iterativa en cada ciclo de entrenamiento [7].

Este algoritmo puede dividirse en las siguientes fases:

- Propagación hacia adelante

  En este paso, los datos de entrada se procesan a través de todas las capas de la red. En cada capa, se aplica una transformación mediante una multiplicación ponderada y una función de activación no lineal. El resultado de este proceso es una salida predicha por la red.

- Cálculo de Función de Pérdida

  Una función de pérdida se define como una composición de funciones que miden la diferencia entre la salida predicha por la red y la salida real. Esto cuantifica el error de las predicciones generadas por la red.

- Propagación hacia atrás

  Como primer punto se calcula la gradiente de la pérdida con respecto a la salida de la red. Luego se propagan dichos gradientes hacia atrás, empezando desde la capa de salida hacia la capa de entrada.

- Actualización de pesos Por último se ajustan los pesos de la red con respecto a los gradients calculados. Esto generalmente se realiza utilizando un algoritmo de optimización de descenso por gradiente.

4.6. Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales son un tipo de red neuronal profunda eficaz para analizar datos con una estructura de cuadrícula, como las imágenes. Las CNN utilizan capas convolucionales que aplican filtros a los datos de entrada para capturar jerarquías y patrones espaciales, seguidas de capas de agrupación que reducen la dimensionalidad y preservan características importantes. Esta arquitectura permite a las CNN aprender y extraer automáticamente características relevantes de las imágenes, lo que las hace altamente eficientes para tareas visuales en comparación con las redes neuronales artificiales tradicionales [17].

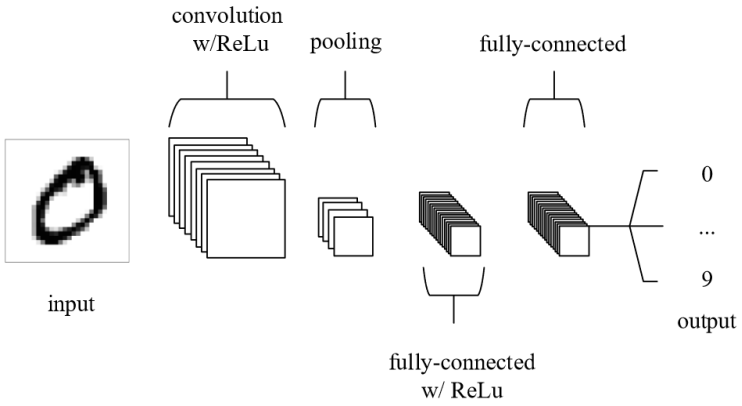

Figura 4: Aqrquitectura simple de una CNN.

[14]

Estas redes están compuestas por tres tipos de capas que trabajan en conjunto: las capas convolucionales, las capas de *"max poolingz* las capas completamente conectadas [17]. A continuación, se describe cada una de estas capas.

4.6.1. Capa convolucional

Una capa convolucional es un componente esencial de las Redes Neuronales Convolucionales que lleva a cabo la operación de convolución sobre los datos de entrada. Mediante el uso de filtros aprendibles, esta capa extrae características importantes de los datos, como bordes y texturas, que son fundamentales para identificar patrones en las imágenes. Los mapas de características generados permiten destacar información relevante en las imágenes, lo que facilita el aprendizaje de la red. Además, la capa convolucional normalmente aplica una función de activación para introducir no linearidad en las representaciones del modelo. Esta capa desempeña un papel clave en la capacidad de las CNNs para aprender representaciones jerárquicas de los datos, lo que optimiza su rendimiento en tareas de análisis y reconocimiento de imágenes [17].

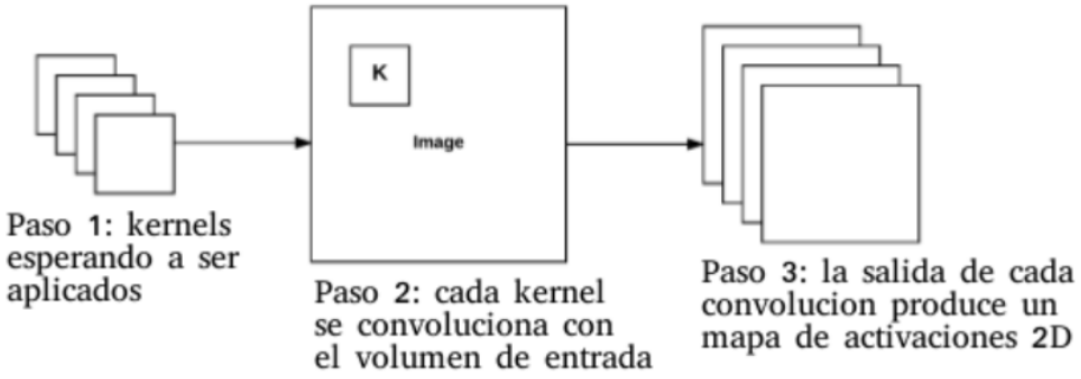

Figura 5: Uso de kernels en el proceso de convolución

[16]

kernel

El kernel también conocido como filtro, es una pequeña matriz que se utiliza para realizar operaciones de convolución en datos de entrada, como imágenes. El núcleo se desliza sobre la imagen de entrada, calculando el producto escalar entre los valores del núcleo y los valores de píxeles correspondientes en la imagen en cada posición. Este proceso genera un mapa de características que resalta características o patrones específicos dentro de la imagen, como bordes, texturas o formas. El tamaño del kernel determina la escala de las características que puede detectar; Los núcleos más pequeños capturan detalles finos, mientras que los kernels más grandes pueden capturar patrones más amplios. Al aprender los valores óptimos del kernel durante el proceso de entrenamiento, la CNN puede extraer de manera efectiva características relevantes de los datos de entrada, lo que le permite reconocer y clasificar imágenes con alta precisión. La operación de convolución, facilitada por el kernel, es esencial para construir representaciones jerárquicas de los datos, lo que permite a la red aprender patrones y relaciones complejos dentro de la entrada [17].

Al realizar la operación de convolución en las Redes Neuronales Convolucionales (CNNs), hay dos parámetros principales que determinan cómo actúa el *kernel* sobre los datos de entrada: el *stride* y el *padding*. Estos parámetros influyen directamente en cómo se mueven los *kernels* a través de la imagen y cómo se manejan los bordes de la misma, afectando tanto la dimensionalidad de los mapas de características resultantes como el nivel de detalle que el modelo puede capturar. A continuación, se explican en detalle ambos parámetros [17].

- **Stride**: El *stride* determina el tamaño del paso con el que el *kernel* se desplaza a lo largo de la imagen de entrada. Un *stride* de 1 significa que el *kernel* se mueve un píxel a la vez, lo que genera una salida altamente superpuesta y produce un mapa de características más grande. Por el contrario, un *stride* mayor (por ejemplo, de 2 o más) reduce la superposición, lo que da lugar a un mapa de

características más pequeño. Esta reducción en tamaño puede ayudar a disminuir la carga computacional y el uso de memoria, pero también puede resultar en la pérdida de información espacial detallada. Por lo tanto, la elección del *stride* afecta tanto la dimensionalidad de la salida como el nivel de detalle capturado por la capa convolucional [17].

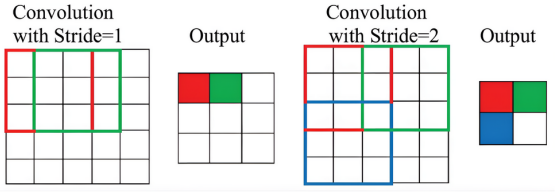

Figura 6: Ilustración de convoluciones con diferentes valores de *stride*

.

[20]

- **Padding**: El *padding* consiste en agregar píxeles adicionales alrededor de los bordes de la imagen de entrada antes de aplicar el *kernel*. Esta técnica ayuda a controlar las dimensiones espaciales del mapa de características de salida. El *zero-padding*, donde los píxeles agregados se establecen en cero, permite que el *kernel* cubra completamente los bordes de la entrada, asegurando que no se ignoren características importantes cercanas a los límites. Sin *padding*, el tamaño de la salida puede disminuir con cada capa convolucional, lo que potencialmente conlleva una pérdida de información. Al ajustar la cantidad de *padding*, los practicantes pueden mantener las dimensiones originales de la entrada o lograr tamaños de salida específicos, influyendo así en la arquitectura y el rendimiento general de la CNN [17].

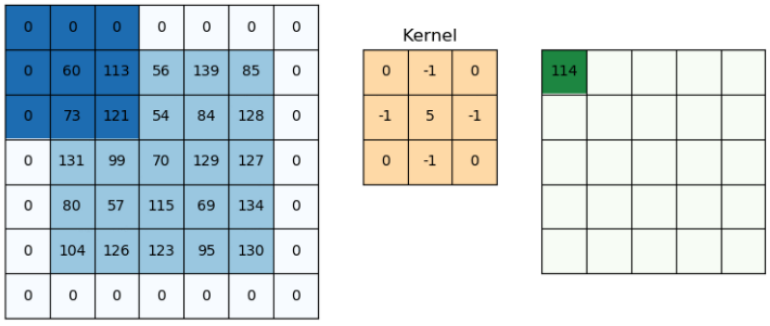

Figura 7: Ejemplo de Zero Padding.

[8]

4.6.2. Capa de Pooling

Pooling es una operación fundamental en las Redes Neuronales Convolucionales (CNNs) que juega un papel crucial en la reducción de las dimensiones espaciales de los mapas de características mientras se conserva la información importante. A este proceso se le conoce como subsampling o downsampling, la técnica de pooling se utiliza para disminuir el tamaño de los mapas de características, lo que ayuda a controlar la complejidad del modelo, reducir el sobreajuste (*overfitting*) y mejorar la eficiencia computacional al disminuir el número de parámetros y el cómputo necesario en las capas subsiguientes [13].

La operación de pooling implica deslizar un filtro bidimensional sobre cada canal del mapa de características y resumir los valores dentro de la región cubierta por el filtro, lo que resulta en un mapa de características reducido pero representativo de los patrones clave de la imagen [13].

Max Pooling

En esta técnica, conocida como *max pooling*, se selecciona el valor máximo dentro de una región específica del mapa de características. Al elegir el valor máximo en cada región, se garantiza que los elementos más destacados o prominentes de la imagen sean conservados en el nuevo mapa de características. De esta manera, se logra reducir las dimensiones espaciales del mapa original, disminuyendo la cantidad de información redundante y al mismo tiempo manteniendo las características más importantes. Este enfoque es útil para hacer el modelo más eficiente en términos computacionales, ya que reduce la cantidad de parámetros en las capas subsiguientes, y también puede ayudar a mejorar la capacidad de generalización del modelo, evitando que se enfoque en detalles poco relevantes de los datos de entrada [13].

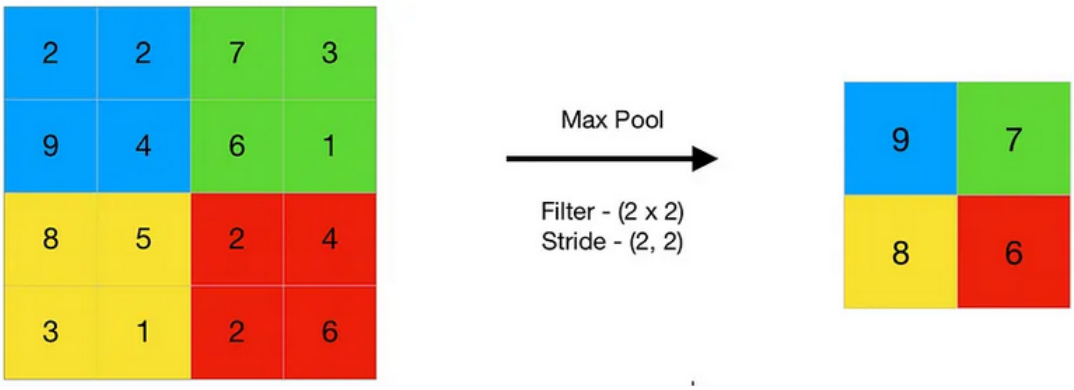

Figura 8: Ejemplo de Max Pooling

.

[13]

Average Pooling

En la técnica conocida como *average pooling*, se calcula el valor promedio de los elementos presentes en una región específica del mapa de características cubierta por el filtro. A diferencia de *max pooling*, que selecciona el valor más prominente en cada área, *average pooling* toma en cuenta todos los valores dentro de la región y calcula su promedio, lo que resulta en un mapa de características más suavizado. Esta técnica permite capturar información generalizada de los datos, reduciendo la sensibilidad a valores extremos o picos específicos y conservando un contexto más equilibrado de la imagen. Aunque disminuye la dimensionalidad del mapa de características, *average pooling* tiende a conservar más información del conjunto completo de valores, en lugar de destacar solo los más dominantes [13].

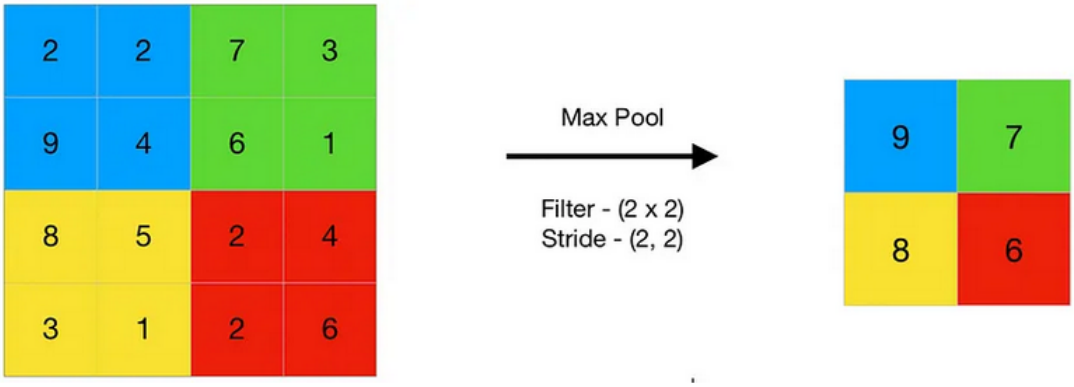

Figura 9: Ejemplo de *Average Pooling*

[13]

4.7. Arquitecturas de Redes Neuronales Convolucionales

La evolución de las arquitecturas de Redes Neuronales Convolucionales (CNN) ha transformado significativamente el panorama de la visión por computadora y la inteligencia artificial. Desde los diseños pioneros que sentaron las bases para la clasificación de imágenes hasta los modelos sofisticados que dominan las aplicaciones contemporáneas, cada arquitectura ha introducido innovaciones únicas con el objetivo de mejorar el rendimiento y la eficiencia. Esta sección explorará la progresión de diversas arquitecturas de CNN, como AlexNet, VGG e Inception, destacando sus características arquitectónicas, los requisitos computacionales y los compromisos necesarios para optimizarlas para su implementación en dispositivos de borde. Al examinar estos avances, podemos obtener una comprensión más profunda de cómo cada modelo ha contribuido a la búsqueda continua de mayor precisión y eficiencia en las tareas de aprendizaje automático [9].

A continuación, se mencionan algunas de las arquitecturas más destacadas de los

últimos años, las cuales han tenido un impacto significativo en el desarrollo y avance de las redes neuronales convolucionales y sus aplicaciones en el ámbito de la inteligencia artificial.

4.7.1. VGG

La arquitectura VGG es un tipo de red neuronal convolucional profunda diseñada para tareas de clasificación de imágenes, caracterizada por el uso de pequeños filtros convolucionales (3×3) apilados en una configuración profunda. VGG enfatiza una arquitectura uniforme donde varias capas convolucionales son seguidas por capas de *max pooling* para reducir progresivamente las dimensiones espaciales. La red culmina en capas completamente conectadas que realizan la clasificación en función de las características extraídas de las capas convolucionales. VGG es conocida por su profundidad, con configuraciones que van desde 11 hasta 19 capas de pesos, lo que le permite aprender características jerárquicas complejas a partir de imágenes [19].

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figura 10: Descripción de configuraciones de la arquitectura VGG.

[19]

4.7.2. ResNet (Red Residual)

ResNet, o Red Residual, es un tipo de arquitectura de red neuronal profunda que utiliza el aprendizaje residual para facilitar el entrenamiento de redes muy profundas. Introduce conexiones de atajo, o conexiones de salto, que permiten que el gradiente fluya a través de la red de manera más efectiva durante la retropropagación. Este diseño ayuda a mitigar el problema de degradación, donde las redes más profundas rinden peor que las más superficiales. Al aprender funciones residuales con referencia a las entradas de la capa, las ResNets pueden lograr una mayor precisión y una mejor optimización, lo que permite la construcción de redes con cientos o incluso miles de capas, manteniendo una complejidad menor en comparación con arquitecturas tradicionales como las redes VGG [11].

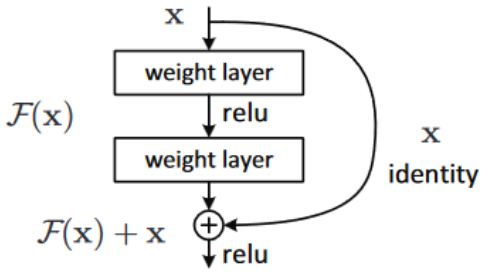

Figura 11: Bloque residual.

[11]

Conexiones Atajo (*shorcut connections*)

La característica principal de ResNet son las conexiones de atajo que saltan una o más capas. Estas conexiones suman la entrada de un bloque a su salida, lo que permite a la red aprender funciones residuales. En lugar de aprender directamente la salida deseada, la red se enfoca en aprender la diferencia (o el residuo) entre la entrada y la salida. Este enfoque ayuda a mitigar el problema del gradiente desvaneciente y facilita el entrenamiento de redes mucho más profundas [11].

Función Residual

Al momento de unir la salida del bloque convolucional con la conexión se utiliza la siguiente ecuación para aproximar la función residual.

$$F(x) = H(x) - x$$

Donde:

- $H(x)$ es la función de mapeo subyacente que se desea aprender.

- x es la entrada a la red.

La idea detrás de la función residual es que, en lugar de aprender directamente la función H(x), la red aprende a aproximar la función residual F(x). Esto permite que la red se enfoque en aprender las diferencias (o residuos) entre la entrada y la salida deseada, lo que facilita la optimización y mejora el rendimiento [11].

4.8. Procesamiento de Lenguaje Natural (NLP)

El procesamiento de lenguaje natural (NLP) es una disciplina de la inteligencia artificial que busca desarrollar métodos para que las computadoras comprendan y extraigan información útil de textos en lenguaje humano. Este campo, que combina elementos de lingüística y ciencia de datos, ha cobrado gran relevancia con el auge de las redes sociales y la generación masiva de datos textuales. Para analizar estos datos, NLP utiliza técnicas como la tokenización, lematización y el uso de diccionarios especializados para emoticones y abreviaturas, facilitando así el procesamiento de expresiones textuales complejas y variadas [4].

Aplicado al análisis de sentimientos y detección de temas, NLP permite identificar emociones y categorizar tópicos en mensajes cortos, superando retos como la brevedad y falta de contexto de estos textos. Estas técnicas también se adaptan al lenguaje particular de las redes sociales mediante el uso de modelos de clasificación y análisis de palabras clave que optimizan la precisión [4].

4.9. Explainable AI (XAI)

La inteligencia artificial explicable (XAI, por sus siglas en inglés) se refiere a métodos y técnicas en inteligencia artificial que hacen que los resultados de los sistemas de IA sean comprensibles para los humanos. El objetivo de XAI es crear modelos que no solo proporcionen predicciones precisas, sino que también ofrezcan información sobre cómo se realizan esas predicciones. Esto implica brindar explicaciones que sean interpretables, transparentes y justificables, permitiendo a los usuarios comprender el razonamiento detrás de las decisiones de la IA [15].

Los aspectos clave de la inteligencia artificial explicable incluyen:

1. **Transparencia**: Los procesos y algoritmos utilizados por la IA deben ser claros y accesibles, permitiendo a los usuarios entender cómo funciona el modelo.
2. **Interpretabilidad**: Las predicciones del modelo deben presentarse de manera comprensible para los usuarios, a menudo a través de visualizaciones o representaciones simplificadas del proceso de toma de decisiones.

3. **Justificabilidad**: Las explicaciones proporcionadas deben estar fundamentadas en los datos y la lógica utilizada por el modelo, permitiendo a los usuarios evaluar la validez de las conclusiones de la IA.
4. **Diseño centrado en el usuario**: Las explicaciones deben adaptarse a las necesidades y al nivel de comprensión de los usuarios finales, sean estos expertos en el área o personas sin conocimientos específicos.
5. **Confianza y responsabilidad**: Al proporcionar explicaciones claras, XAI busca generar confianza en los sistemas de IA y hacerlos responsables de sus decisiones, lo cual es particularmente importante en aplicaciones de alto riesgo como la atención médica, las finanzas y el derecho.

4.9.1. SHAP

SHAP (SHapley Additive exPlanations) es un enfoque basado en teoría de juegos para explicar los resultados de cualquier modelo de aprendizaje automático. Esta técnica vincula la asignación óptima de créditos con explicaciones locales, utilizando los valores de Shapley de la teoría de juegos y sus extensiones relacionadas [15].

Shapley values

Los valores de Shapley proporcionan un método estructurado para distribuir de manera justa las ganancias totales entre los jugadores de una coalición, en función de la contribución individual de cada jugador al resultado global [15]. Este concepto se ha convertido en fundamental en campos como la economía, la ciencia política y el aprendizaje automático, donde se utiliza ampliamente para interpretar las contribuciones individuales de las características en las predicciones de modelos [15].

Los valores de Shapley son especialmente útiles en aplicaciones que requieren una distribución justa de recursos o responsabilidades. En economía, se utilizan para analizar la distribución de costos y ganancias; en ciencia política, para medir el poder de voto; y en el aprendizaje automático, para ayudar a explicar las predicciones de los modelos al asignar una puntuación de contribución a cada característica [15]. Calcular los valores de Shapley puede ser computacionalmente complejo para juegos grandes, lo que ha llevado al desarrollo de métodos de aproximación como SHAP (SHapley Additive exPlanations) en aplicaciones prácticas de aprendizaje automático, donde mejoran la transparencia del modelo al indicar la influencia de cada característica en la predicción [15].

4.9.2. Computación paralela

4.10. Virtualización

CAPÍTULO 5

Marco Metodológico

5.1. Detección de Rostros

Para el proceso de detección de rostros en tiempo real, se emplearon los detectores en cascada de la biblioteca OpenCV [6]. Específicamente, se utilizó el módulo textit{CascadeClassifier} junto con los filtros de Haar proporcionados por OpenCV, en particular, el archivo preentrenado *haarcascade_frontalface_default.xml}.* Estos detectores se caracterizan por ser eficaces para la detección de objetos debido a su estructura jerárquica y secuencial, que permite clasificar subventanas en una imagen de manera rápida y eficiente, descartando aquellas que no contienen el objeto de interés en las primeras etapas.

La elección de este método se fundamenta en la alta eficiencia computacional de los clasificadores en cascada. En el artículo "*Rapid Object Detection using a Boosted Cascade of Simple Features*", los autores destacan que, con un conjunto de datos de 507 imágenes y 75 millones de subventanas, el tiempo de detección de rostros fue aproximadamente 15 minutos menor en comparación con métodos basados en redes neuronales [21]. Esto resulta especialmente útil para este proyecto, ya que la detección de rostros es solo la primera etapa de un pipeline que se pretende ejecutar en tiempo real.

Con base en la descripción anterior se implementó un detector de cascada para detectar rostros con las siguientes configuraciones:

1. Se convirtió la imágen de entrada a escala de grises utilizando la función *cvtColor*.
2. Se aplica el método `detectMultiScale` con los siguientes parámetros:

a) ***scaleFactor* = 1.3:** Este parámetro determina cuánto se reduce el tamaño de la imagen en cada escala. En este caso, se utilizó un valor de 1.3 en lugar del valor predeterminado de 1.1, lo cual permite evaluar escalas más variadas de manera más eficiente. Aunque valores más bajos pueden mejorar la precisión, también aumentan la complejidad computacional. Con este valor, se logró un equilibrio adecuado entre precisión y velocidad en la detección.

b) ***minNeighbors* = 5:** Este parámetro define el número mínimo de vecinos que debe tener una subventana para ser considerada una detección válida. Un valor alto ayuda a prevenir falsos positivos, aunque podría ralentizar el proceso de predicción. Por otro lado, valores más bajos podrían incrementar la incidencia de falsos positivos.

c) ***minSize* = 48:** Este parámetro especifica el tamaño mínimo que debe tener el objeto para ser detectado. Al establecer un valor de 48, se ignoran rostros más pequeños que 48x48 píxeles, lo cual es útil para evitar la detección de detalles irrelevantes en escenas complejas o de alta resolución. Este valor se eligió porque el modelo de clasificación de expresiones faciales trabaja con imágenes de 48x48; por lo tanto, detectar rostros de menor tamaño sería contraproducente.

5.2. Analisis de Sentimiento en rostros

5.2.1. Obtención de Datos

Para entrenar un modelo capaz de clasificar expresiones faciales, fue necesario disponer de un conjunto de datos que incluyera ejemplos de diversas expresiones faciales previamente etiquetadas. Además, se planteó el reto de optimizar el modelo para que realizara predicciones de manera eficiente y en tiempo real, lo que llevó a seleccionar el dataset *FER+*. Este conjunto de datos se deriva del *FER2013*, un dataset creado en 2013 que contenía aproximadamente 30,000 imágenes de rostros de 48x48 píxeles, clasificadas en siete expresiones: enojo, disgusto, miedo, felicidad, tristeza, sorpresa y neutral. Sin embargo, el *FER2013* presentaba un problema significativo de clasificaciones erróneas. En 2016, Microsoft realizó un proceso de reetiquetado para mejorar la precisión del dataset y optimizar el rendimiento de los modelos entrenados con este.

El proceso de etiquetado del dataset FER+ se realizó mediante *crowdsourcing*, donde cada imagen fue etiquetada por 10 personas distintas. A los etiquetadores se les pidió clasificar cada imagen en una de ocho emociones: neutral, felicidad, sorpresa, tristeza, ira, disgusto, miedo y desprecio. Inicialmente, se buscó asegurar la calidad del etiquetado mediante un método de estándar de oro, deteniendo el proceso una vez que dos etiquetadores coincidían en una sola emoción; sin embargo, esta primera aproximación no generó la calidad esperada [5].

Para mejorar la precisión, los investigadores decidieron que las 10 personas etiquetaran cada imagen, obteniendo así una distribución de emociones para cada rostro.

La etiqueta mayoritaria entre los 10 etiquetadores fue considerada una buena aproximación. Además, el estudio implementó un mecanismo de control de calidad del etiquetado, mostrando que un mayor número de etiquetadores impactaba positivamente en la calidad de la etiqueta final, con mayores tasas de concordancia a medida que aumentaba el número de participantes [5].

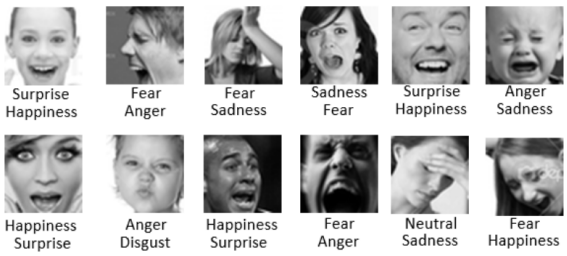

Figura 12: Ejemplos de etiquetas de FER y FER+ (Las etiquetas superioes pertenecen a FER y las inferiores a FER+).

[5]

Este dataset resultó especialmente útil para el proyecto debido a que sus imágenes son de tamaño reducido (48x48 píxeles) y están en escala de grises, lo cual reduce significativamente la carga computacional en comparación con imágenes grandes y a color (RGB), algo crucial para aplicaciones en tiempo real. Además, el *FER+* es considerablemente más preciso que su predecesor *FER2013*, debido al proceso de reetiquetado.

5.2.2. Preparación de Ambiente para Entrenamiento de modelo

Se decidió utilizar PyTorch como el *framework* principal para el entrenamiento y desarrollo del modelo, debido a su flexibilidad, facilidad de uso y amplio soporte para operaciones en GPU. Como entorno de pruebas, exploración de datos y entrenamiento de modelos, se utilizó Docker para asegurar la portabilidad y reproducibilidad del modelo en diferentes plataformas. Para esto se utilizó la imágen *pytorch:2.4.0-cuda11.8-cudnn9-runtime*, la cual incluye soporte para CUDA. Esto permite aprovechar la GPU del sistema anfitrión, en caso de estar disponible, para acelerar el procesamiento. Esto resulta especialmente útil al trabajar con imágenes, ya que reduce de forma significativa el tiempo de entrenamiento de los modelos en comparación con el uso exclusivo de CPU.

La configuración incluyó la creación de un contenedor basado en esta imagen de Docker, el cual se levantó con un servidor de Jupyter mapeado al puerto 8888 para facilitar el acceso y la ejecución de notebooks. Esto permitió trabajar de forma interactiva con el kernel de Python optimizado para GPU, lo cual es esencial para ajustar y probar diferentes configuraciones de modelo de manera eficiente.

Además, se establecieron volúmenes compartidos entre el contenedor y el sistema

anfitrión para facilitar el acceso a los datos y permitir la persistencia de resultados sin necesidad de reconstruir el contenedor en cada sesión. La combinación de Docker y Jupyter también facilitó el ajuste de parámetros, la monitorización del rendimiento del modelo y la visualización de los resultados de entrenamiento en tiempo real. Esta configuración asegura un flujo de trabajo flexible y reproducible, permitiendo una fácil integración y despliegue del modelo en distintos entornos de producción o investigación.

5.2.3. Preparación de datos

Como primer paso, se realizó una exploración exhaustiva de los datos, con el fin de comprender su estructura y características generales. Durante esta fase, se observó que el dataset presentaba un desbalance significativo en las clases. La clase más representada contaba con aproximadamente 10,000 ejemplos, mientras que la clase menos representada solo tenía 165 ejemplos. como se muestra en la siguiente figura:

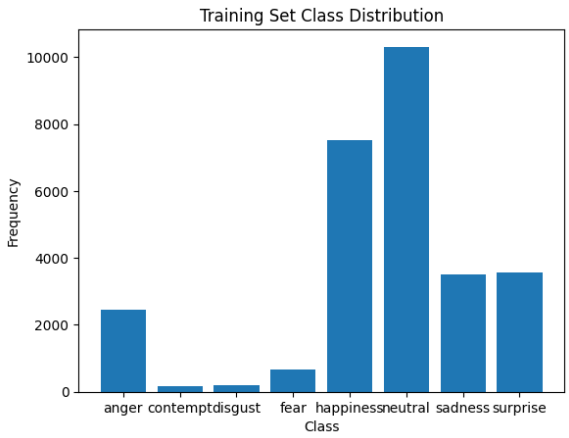

Figura 13: Distribución de conteo de clases para el conjunto de datos FER+.

Este desbalance implica un riesgo de sesgo en el modelo, ya que puede tender a favorecer la clase dominante, afectando negativamente la capacidad del modelo para reconocer correctamente las clases minoritarias. Por tanto se eliminar las clases "*fear*", "*contemptz* "*disgust*". El conteo en conjunto de las clases eliminadas representa el 3.55 % de las muestras del conjunto de datos. lo cual hace que su eliminación tenga un impacto mínimo en la variabilidad general del conjunto de datos, pero contribuye a mejorar la robustez del modelo. Sin embargo luego La distribución resultante luego de la eliminación se presenta a continuación.

Figura 14: Distribución de conteo de clases para el conjunto de datos FER+ sin las clases "*fear*", "*contemptz* "*disgust*".

Luego de la eliminación se presenta una diferencia menor con la clase más conteo con respecto a la clase con menor conteo. Sin embargo, esto no fue suficiente para lograr un balanceo en el conjunto de datos. Para esto se implementó una técnica de submuestreo en dónde se recortan las clases para igualar el conteo de la clase con menor conteo. A continuación se muestra la distribución del conteo de datos luego del submuestreo.

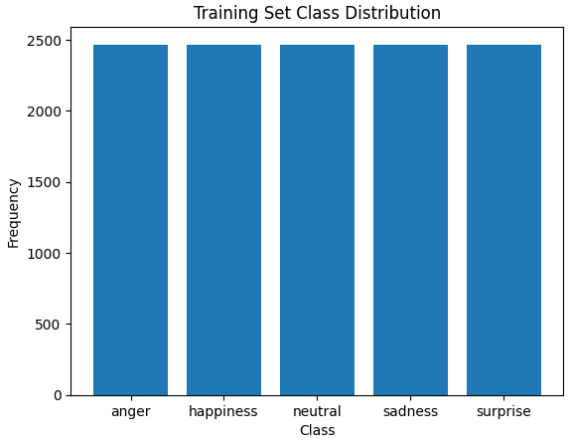

Figura 15: Distribución de conteo de clases para el conjunto de datos FER+ con submuestreo.

Con esto se logró un dataset balanceado con 2466 muestras para cada clase.

Como segundo punto, se definieron transformaciones específicas tanto para el conjunto. Estas transformaciones ayudan a estandarizar y aumentar los datos de entrada, optimizando el modelo para un mejor rendimiento en la clasificación de expresiones faciales.

1. **Transformaciones para el Conjunto de Entrenamiento y Validación**:
   - Se convirtieron las imágenes a escala de grises para reducir la dimensionalidad y adaptar los datos al modelo.
   - Las imágenes se redimensionaron a 48x48 píxeles, en concordancia con el tamaño utilizado en el entrenamiento de modelos similares.
   - Se aplicó una equalización de histograma para mejorar el contraste de las imágenes, destacando características faciales importantes y reduciendo la variabilidad causada por diferencias en iluminación.
2. **Transformaciones para el Conjunto de Entrenamiento**:
   - Para aumentar la diversidad de datos, se implementaron transformaciones aleatorias:
     - Una probabilidad del 50 % de voltear horizontalmente cada imagen.
     - Una rotación aleatoria de hasta 15 grados.
3. **Carga de Datos y Submuestreo**:
   - Los conjuntos de entrenamiento y validación se cargaron usando la clase `ImageFolder` de PyTorch, que organiza las imágenes según sus carpetas de clase.
   - Además, se desarollo una función de submuestreo, que permite equilibrar los datos de las diferentes clases. Cabe mencionar que este paso solo se lleva a cabo para el conjunto de datos de entrenamiento.
4. **Creación de DataLoaders**: Se crearon `DataLoaders` para cada conjunto de datos, con un tamaño de lote de 64 y configuración para procesamiento paralelo (`num_workers=4`). El `DataLoader` de entrenamiento está configurado para mezclar aleatoriamente (`shuffle=True`) los datos en cada época, mientras que el de validación no realiza mezclado para mantener la consistencia.

5.2.4. Entrenamiento

Para realizar el entrenamiento de los modelos, se diseñó un esquema de entrenamiento que contempla tanto hiperparámetros estáticos como variables, con el objetivo de encontrar la configuración que optimice las métricas de rendimiento del modelo sin incrementar el riesgo de sobreajuste. Los hiperparámetros estáticos fueron seleccionados para mantener consistencia en el proceso de entrenamiento, mientras que los hiperparámetros variables permitieron explorar diferentes niveles de regularización. A continuación, se describen los parámetros utilizados:

- **Función de Pérdida**: Se empleó la función de pérdida `CrossEntropyLoss`, adecuada para tareas de clasificación multiclase, ya que calcula la divergencia entre las predicciones del modelo y las etiquetas verdaderas.

- **Optimizador**: Se utilizó el método de *Stochastic Gradient Descent* (SGD), el cual es ampliamente reconocido por su eficiencia en la optimización de modelos de aprendizaje profundo. Este método actualiza los pesos del modelo de manera estocástica, lo que ayuda a escapar de mínimos locales y acelera la convergencia.
- **Learning Rate**: El valor de la tasa de aprendizaje fue fijado en 0.1, un valor que equilibra la velocidad de convergencia y la estabilidad del entrenamiento, evitando cambios drásticos en los pesos.

Para evaluar el impacto de la regularización en el modelo, se probaron diferentes valores de regularización L1 (peso de decaimiento). La regularización ayuda a controlar el sobreajuste, favoreciendo que el modelo aprenda patrones generales en lugar de memorizar el conjunto de entrenamiento. Los valores explorados fueron:

- **Sin Regularización**: Para obtener una línea base de comparación, se entrenaron modelos sin ninguna regularización L1, lo que permitió observar el comportamiento del modelo sin restricciones en los pesos.
- **Regularización L1 con valor de 0.01**: Se probó este valor para aplicar una penalización moderada a los pesos, lo cual ayuda a limitar la complejidad del modelo sin afectar significativamente su capacidad de ajuste.
- **Regularización L1 con valor de 0.001**: Se probó un valor de penalización más bajo, permitiendo una menor restricción en los pesos y evaluando si un nivel más sutil de regularización ofrece beneficios en la generalización del modelo.

Este esquema de entrenamiento y pruebas permitió comparar los resultados entre modelos con diferentes niveles de regularización, facilitando la selección de hiperparámetros adecuados para optimizar el desempeño y la capacidad de generalización de los modelos con los que se trabajó.

Para el entrenamiento se consideraron las siguientes 4 arquitecturas de red convolucional.

CNN Simple

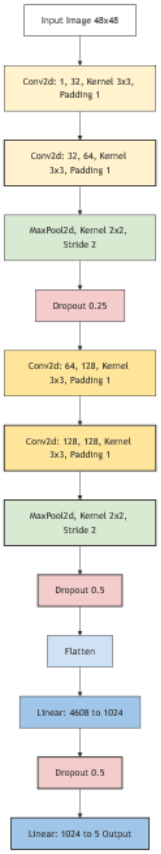

Figura 16: Arquitectura de CNN simple.

Resnet50

Esta arquitectura se centra en las conexiones de atajo. Esto permite entrenar redes más profundas sin aumentar drásticamente la complejidad computacional. Específicamente, ResNet-50 se compone de múltiples bloques residuales. Cada bloque incluye tres capas convolucionales con filtros de 1x1, 3x3 y 1x1 respectivamente. Después de cada convolución, se aplica normalización por lotes y una activación ReLU. Además, cada bloque cuenta con una conexión de atajo que omite las capas convolucionales y suma la entrada del bloque a su salida [11]. En total, ResNet-50 contiene 16 bloques residuales, organizados en cuatro grupos:

- **Grupo 1**: 3 bloques con 64 filtros.
- **Grupo 2**: 4 bloques con 128 filtros.
- **Grupo 3**: 6 bloques con 256 filtros.
- **Grupo4**: 3 bloques con 512 filtros.

PyTorch ofrece una implementación de la arquitectura ResNet-50, la cual permite utilizar este modelo sin necesidad de construirlo desde cero. En este caso, se aprovechó esta implementación preexistente, pero fue necesario realizar algunas modificaciones para adaptarla a las especificaciones del proyecto.

1. En primer lugar, se ajustó la entrada del modelo para que pudiera procesar imágenes en escala de grises de tamaño 48x48, en lugar de las imágenes RGB de 224x224 para las que está diseñado originalmente. Esto implicó modificar la primera capa convolucional para aceptar un solo canal en lugar de tres.
2. Además, se modificó la última capa completamente conectada para que generara una salida correspondiente a 5 clases, adaptando el modelo al conjunto de datos con el que se está trabajando.

Estas adaptaciones nos permitieron utilizar la arquitectura ResNet-50 en un contexto diferente al de su diseño original, maximizando su rendimiento al problema que se está trabajando.

VGG11

VGG11 es una configuración específica de la arquitectura VGG que consta de 11 capas de pesos, incluyendo capas de convolución y capas completamente conectadas. La red se compone de 8 capas convolucionales con filtros pequeños de 3x3, organizadas en bloques seguidos de capas de max pooling para reducir las dimensiones espaciales. Cada capa convolucional utiliza la función de activación ReLU para introducir no linealidad. Después de las capas convolucionales y de pooling. VGG11 cuenta con tres capas completamente conectadas [19]. En este caso se utilizó la implementación que brinda Pytorch de VGG y Al igual que con la arquitectura ResNet-50 fue necesario realizar las siguientes modificaciones para adaptar la arquitectura al problema que se está trabajando:

1. , se ajustó la entrada del modelo para que pudiera procesar imágenes en escala de grises de tamaño 48x48, en lugar de las imágenes RGB de 224x224 para las que está diseñado originalmente. Esto implicó modificar la primera capa convolucional para aceptar un solo canal en lugar de tres.
2. se modificó la última capa completamente conectada para que generara una salida correspondiente a 5 clases, adaptando el modelo al conjunto de datos con el que se está trabajando.

VGG Customizada

Esta arquitectura personalizada surgió al observar que ResNet-50 generaba un sobreajuste pronunciado, mientras que VGG lograba el objetivo de mejor forma. Por

lo tanto, se decidió probar con una arquitectura similar a VGG11 con algunas modificaciones para mitigar el problema de sobreajuste. Esta configuración incluye dos bloques convolucionales similares a los de VGG. El primer bloque cuenta con 3 capas convolucionales, con un total de 256 canales de salida, seguidas de una capa de max pooling y una capa de dropout. El segundo bloque incluye una sola capa convolucional con 256 canales de salida, seguida también de una capa de max pooling y una capa de dropout. Finalmente, la arquitectura incorpora 2 capas completamente conectadas.

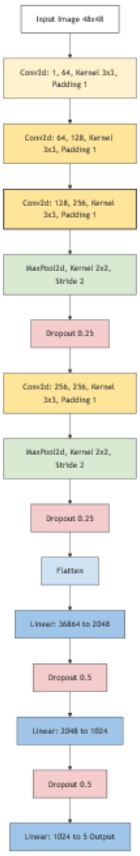

Figura 17: Arquitectura de VGG customizada.

5.2.5. Explicabilidad

Se utilizó SHAP para analizar la explicabilidad del modelo, permitiendo interpretar cómo las características afectan las predicciones. Mediante este enfoque, se generaron valores de importancia que indican la influencia de cada característica, ayudando a entender qué factores son los más relevantes en las decisiones del modelo [15].

5.2.6. Pipeline de Detección de imágenes y clasificación de expresiones faciales

Para lograr la detección de expresiones faciales en tiempo real, se utilizó un detector de rostros previamente implementado, que sirve como entrada para el modelo de clasificación de expresiones faciales. Para ello, fue necesaria la creación de una API REST que recibe una imagen y devuelve la predicción de la expresión facial detectada. Esta API se configuró en un entorno Docker, utilizando como base la imagen "pytorch:2.4.0-cuda11.8-cudnn9-runtime". La API es consumida por un script que, mediante OpenCV, captura la entrada de una cámara en tiempo real, detecta los rostros y luego los procesa a través del modelo de detección de expresiones. Finalmente, cada rostro detectado se enmarca con un cuadro y se muestra en pantalla la expresión correspondiente a cada uno.

5.3. Analisis de sentimiento en Texto

Para el análisis de sentimientos, se utilizó el modelo preentrenado "*distilbert/distilbert-base-uncased-finetuned-sst-2-english* "de Hugging Face. Este modelo está optimizado para el análisis de sentimientos en inglés, proporcionando una base sólida para interpretar y clasificar automáticamente la polaridad de los comentarios en redes sociales y otras fuentes textuales en este idioma. Gracias a su entrenamiento en una amplia variedad de datos en español, el modelo ofrece una alta precisión en la detección de emociones, lo que permite identificar rápidamente la satisfacción, insatisfacción y otros sentimientos expresados por los usuarios en sus interacciones con la marca.

5.3.1. Explicabilidad

Al igual que con el modelo de clasificación de expresiones faciales, se utilizó SHAP para identificar las características que contribuyen a la clasificación final. Esta herramienta permite visualizar el impacto de cada palabra en la predicción, mostrando cuánto influyen de manera positiva o negativa en el resultado final. A continuación se muestran las pruebas realizadas de explicabilidad con SHAP para el modelo de análisis de sentimiento en texto.

5.4. Propuesta para estudios de mercado

La Licenciada Rocío López cuenta con una destacada trayectoria en el ámbito del marketing y la gestión de marcas, con 14 años de experiencia en el desarrollo de estrategias de mercado, posicionamiento de marca, y coordinación de campañas para empresas de renombre en América Latina. Durante la colaboración, se le presentaron los modelos implementados y se demostró su funcionamiento, lo que permitió discutir

y explorar posibles aplicaciones en estudios de mercado. A partir de esta discusión, surgió la idea de aplicar el pipeline de reconocimiento de expresiones faciales para estudios de grupos focales en tiempo real. En esta aplicación, el sistema debe capturar y analizar las expresiones faciales de varios participantes simultáneamente, proporcionando un análisis en tiempo real de sus reacciones. Además, el modelo de análisis de sentimiento puede utilizarse para evaluar los comentarios y respuestas verbales de los participantes, ofreciendo una perspectiva integral sobre sus percepciones y emociones en relación con los temas evaluados en el grupo focal. A continuación, se presenta un diagrama de flujo que describe el uso del los modelos desarrollados para un grupo focal.

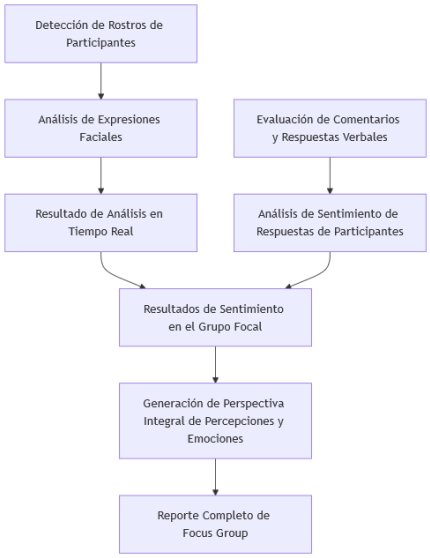

Figura 18: Diagrama de flujo de propuesta de pipeline utilizando el modelo de clasificación de expresiones faciales y el de análisis de sentimiento en texto.

5.5. Pruebas

Pipeline de reconocimiento de expresiones faciales

Para realizar las pruebas del pipeline de reconocimiento de expresiones faciales, se ejecutó el sistema con un grupo aproximado de 10 personas, con el objetivo de validar su funcionalidad en un entorno simulado de grupos focales. Esta decisión se tomó en base a la recomendación del experto en el área, quien indicó que los grupos focales suelen involucrar una cantidad considerable de participantes. De esta manera, el experimento buscó evaluar la capacidad del sistema para identificar y analizar las expresiones faciales de varias personas simultáneamente, garantizando que la herramienta sea adecuada para aplicaciones en contextos reales con un número de usuarios significativo.

Además, se evaluó la eficiencia y rendimiento del programa para asegurar que el

sistema pudiera operar sin interrupciones o disminuciones significativas en su desempeño cuando se enfrentara a un volumen considerable de procesamiento en tiempo real. Para medir la capacidad del programa de mantener un funcionamiento fluido, se realizó un monitoreo constante de los cuadros por segundo (FPS), lo cual permitió observar si la velocidad de procesamiento se mantenía estable durante la prueba con múltiples personas.

CAPÍTULO 6

Resultados

A continuación se presentan los resultados de los entrenamientos par las diferentes arquitecturas y configuraciones.

6.1. CNN Simple

- **Sin regularización.**

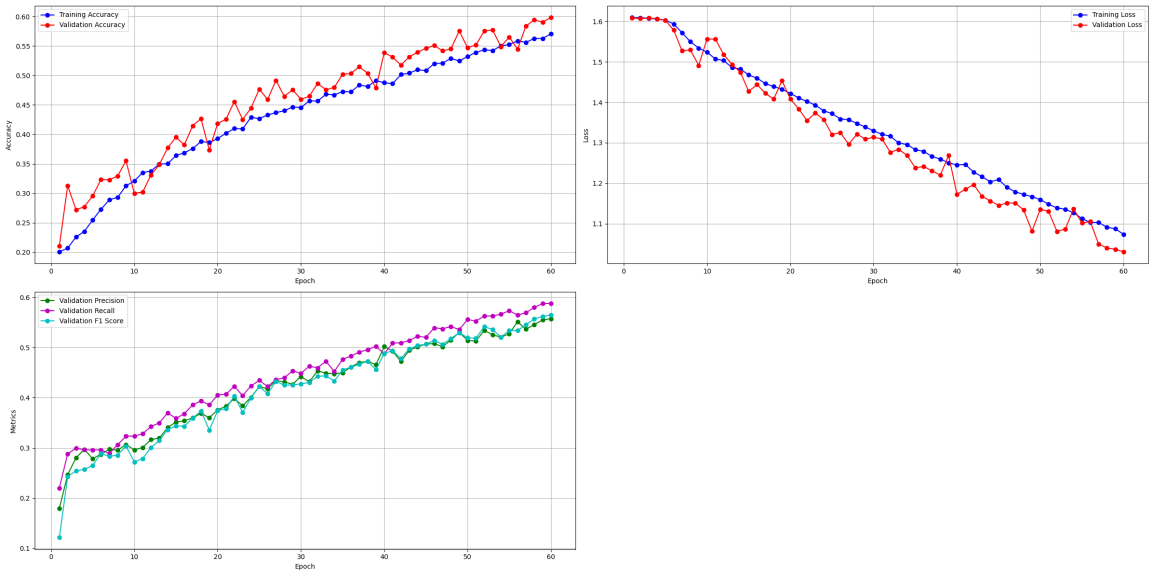

Figura 19: Historial de entrenamiento de CNN simple sin regularización.

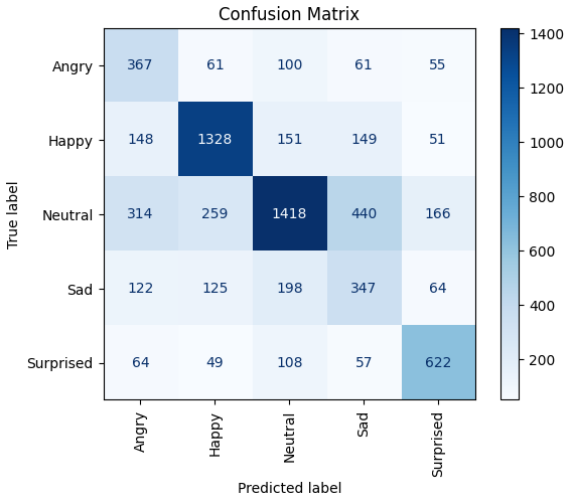

Figura 20: Matriz de confusión para CNN simple entrenada sin regularización.

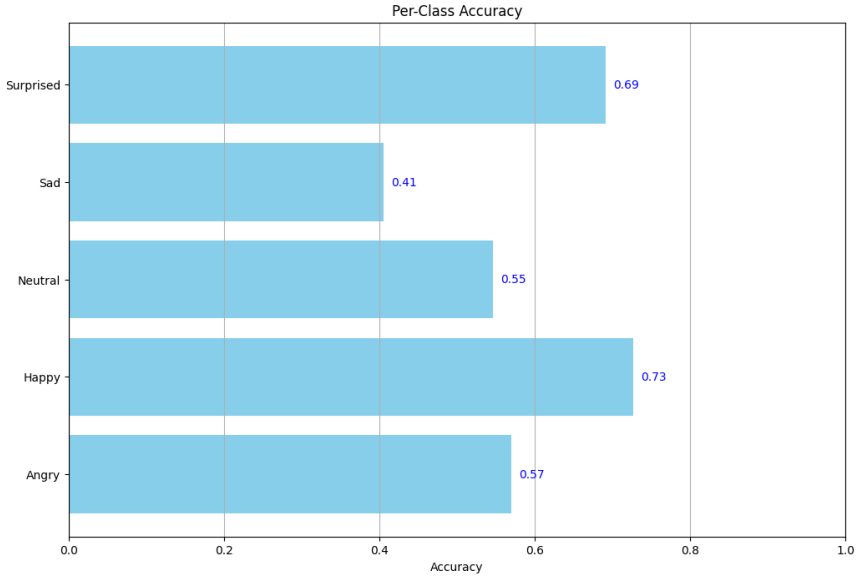

Figura 21: "*Accuracy*"por clase para CNN simple entrenada sin regularización.

- **Regularización L1 con valor de 0.01**.

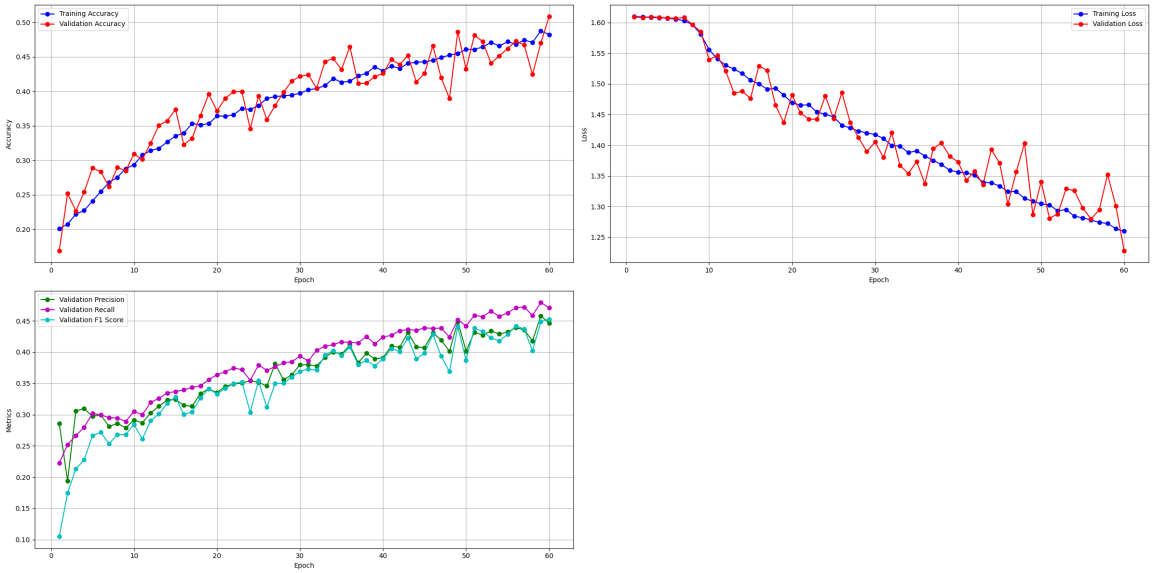

Figura 22: Historial de entrenamiento de CNN simple con regularización L1 con valor de 0.01.

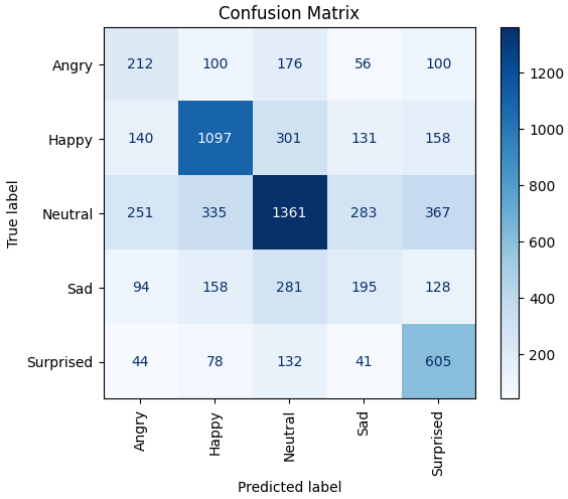

Figura 23: Matriz de confusión para CNN simple con regularización L1 con valor de 0.01.

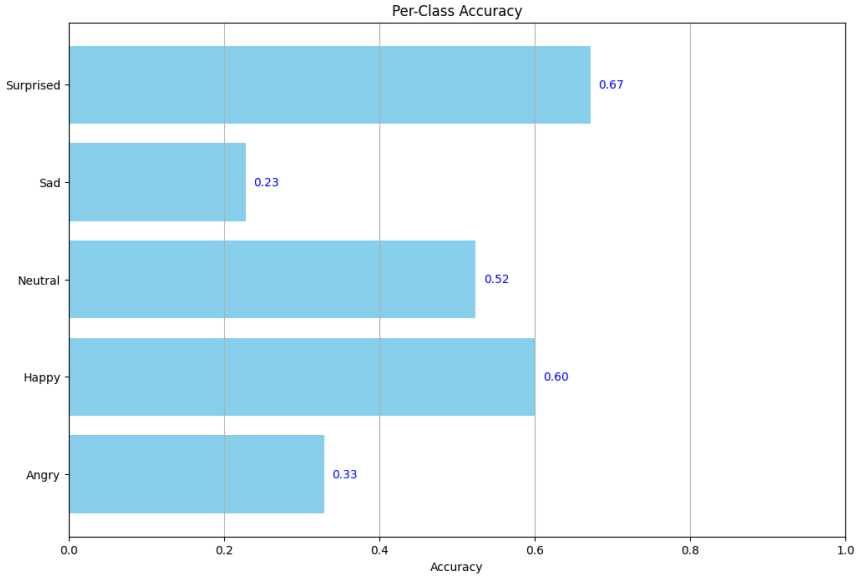

Figura 24: "*Accuracy*"por clase para CNN simple con regularización L1 con valor de 0.01

- **Regularización L1 con valor de 0.001**.

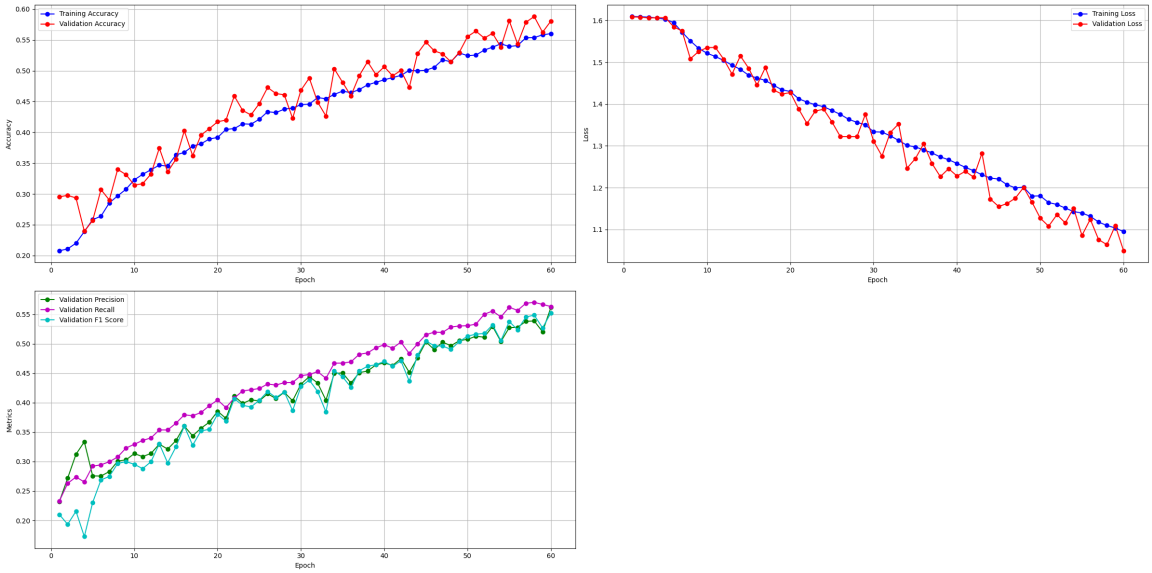

Figura 25: Historial de entrenamiento de CNN simple con regularización L1 con valor de 0.001.

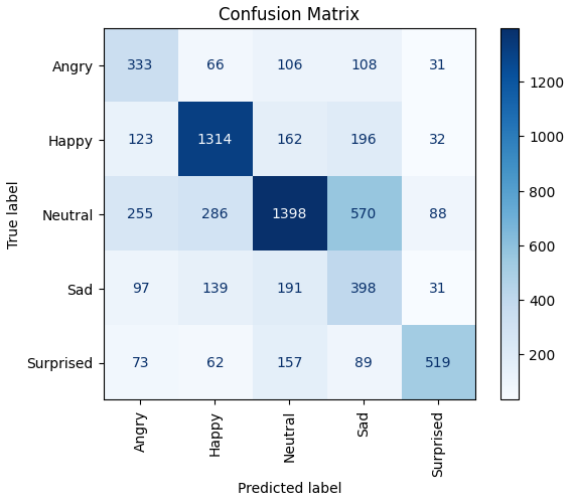

Figura 26: Matriz de confusión para CNN simple con regularización L1 con valor de 0.001.

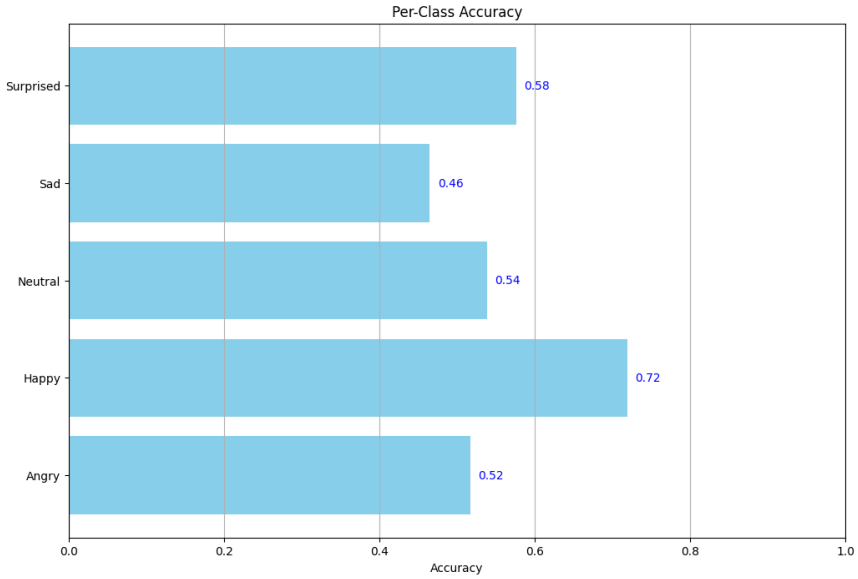

Figura 27: "*Accuracy*"por clase para CNN simple con regularización L1 con valor de 0.001

En las graficas anteriores se puede notar que para las los entrenamientos realizados se tiene una tendencia a clasificar considerablemente mejor las clases "*Happyz* "*Neutral*". Además se puede notar un bajo rendimiento por parte de la arquitectura ya que se obtuvo tener un máximo de 0.56 de F1 score.

6.2. Resnet50

- **Sin regularización.**

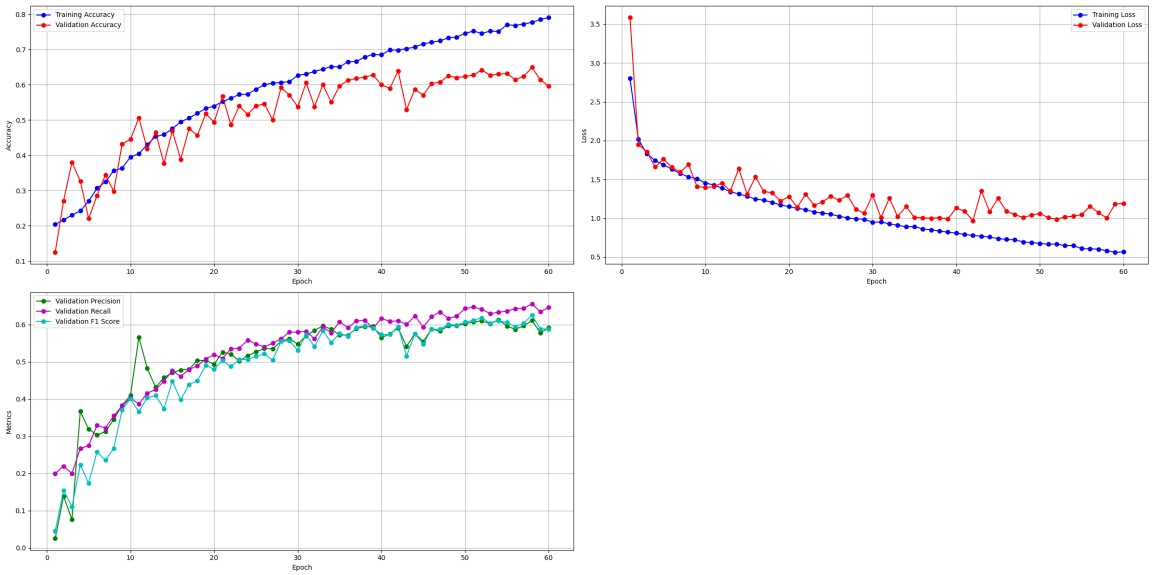

Figura 28: Historial de entrenamiento de ResNet-50 sin regularización.

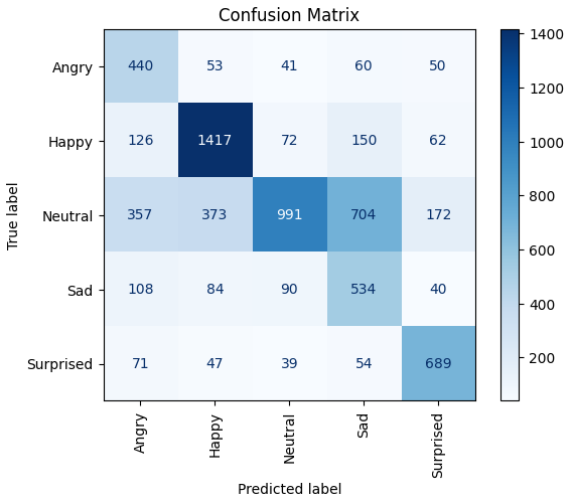

Figura 29: Matriz de confusión para ResNet-50 entrenada sin regularización.

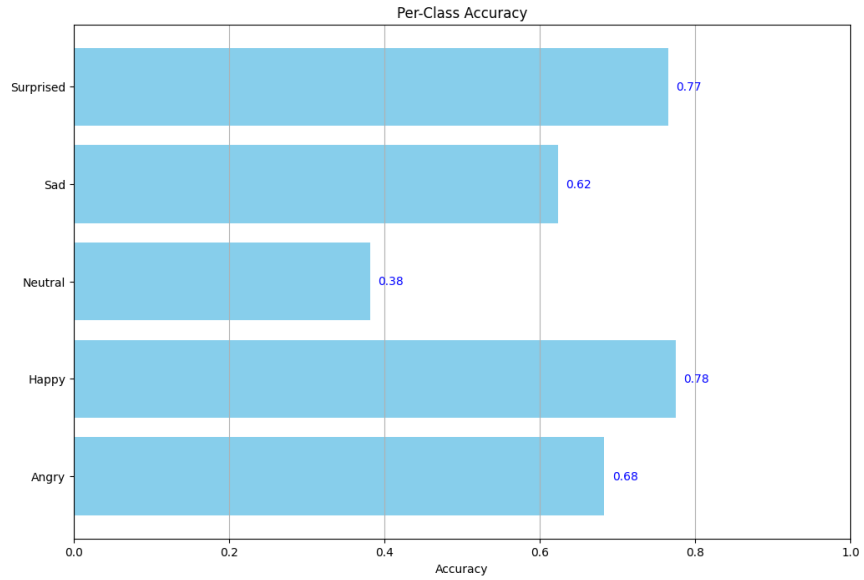

Figura 30: "*Accuracy*"por clase para ResNet-50 entrenada sin regularización.

- **Regularización L1 con valor de 0.01**.

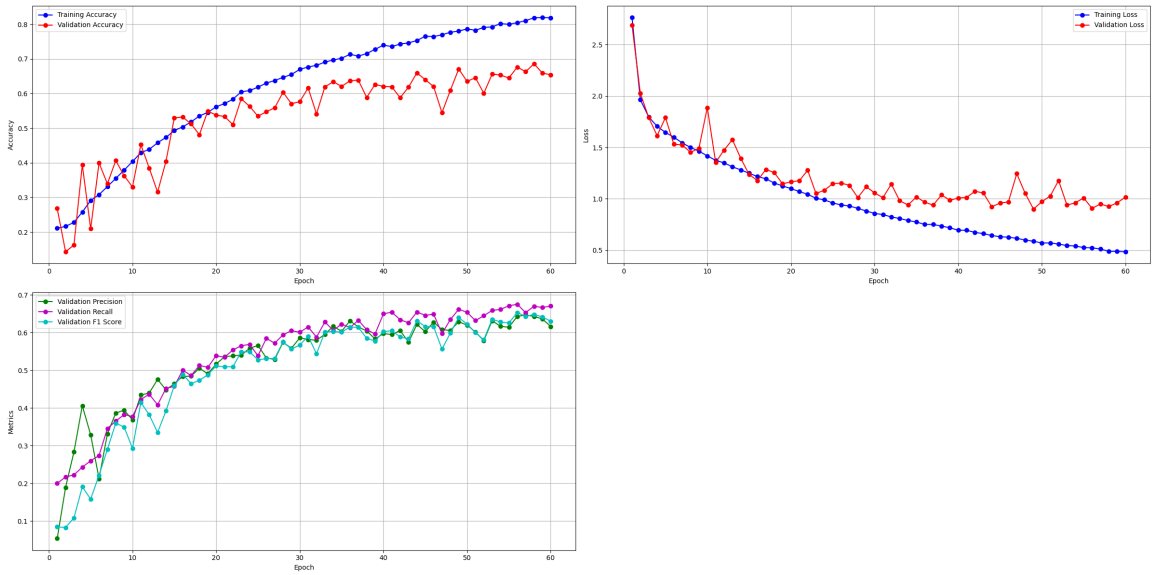

Figura 31: Historial de entrenamiento de ResNet-50 con regularización L1 con valor de 0.01.

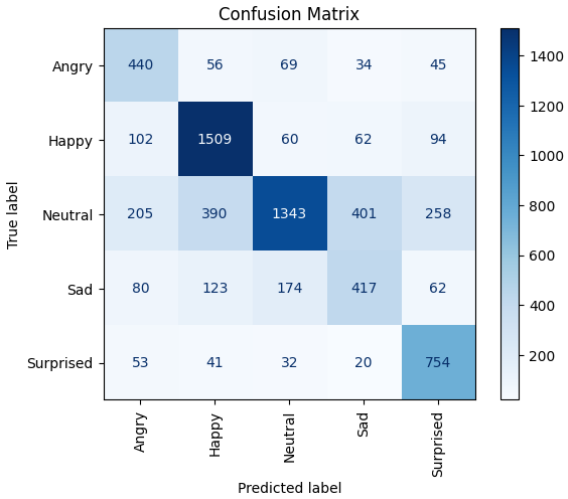

Figura 32: Matriz de confusión para ResNet-50 con regularización L1 con valor de 0.01.

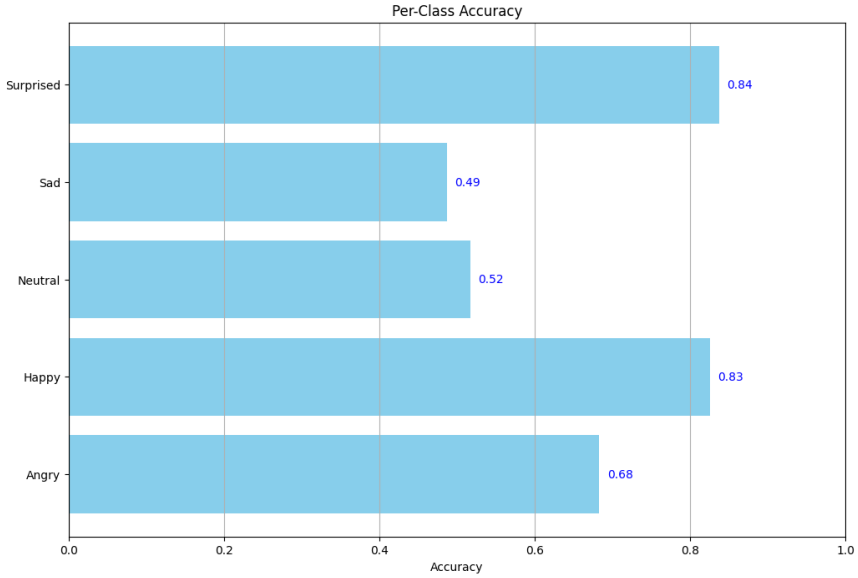

Figura 33: "*Accuracy*"por clase para ResNet-50 con regularización L1 con valor de 0.01

En base a las métricas presentes en las figuras anteriores se puede observar que para esta arquitectura se tienen mejores métricas para las clases "*Happyz* "*Surprised*". Por otro lado en las gráficas de historial de entrenamiento se puede notar sobreajuste ya que para los entrenamientos realizados se puede ver que después de una época determinada el "*accuracy*"de entrenamiento y el de validación divergen.

6.3. VGG11

- **Sin regularización.**

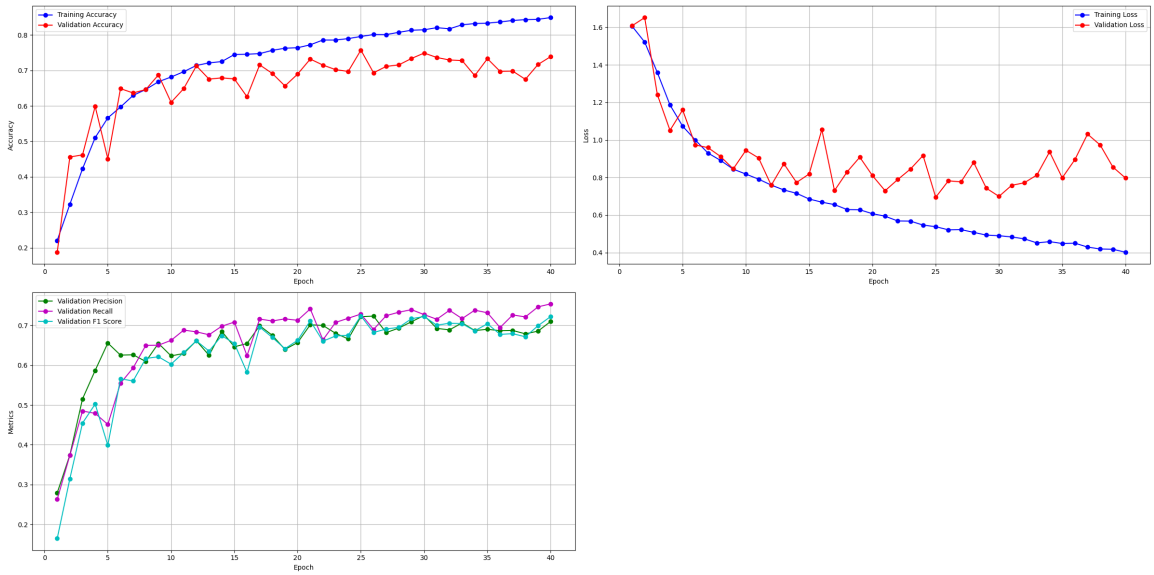

Figura 34: Historial de entrenamiento de VGG11 sin regularización.

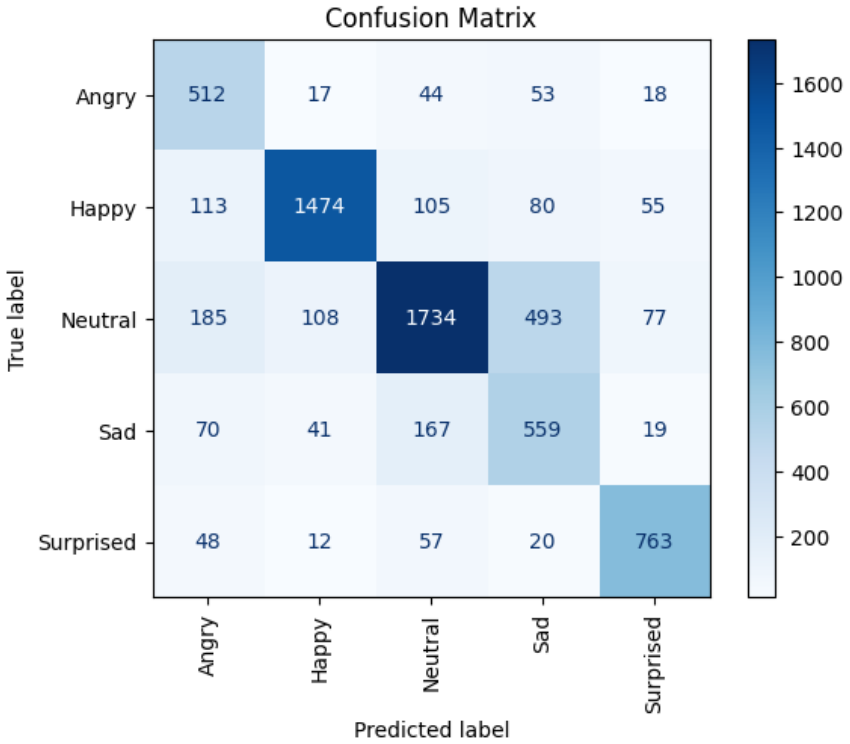

Figura 35: Matriz de confusión para VGG11 entrenado sin regularización.

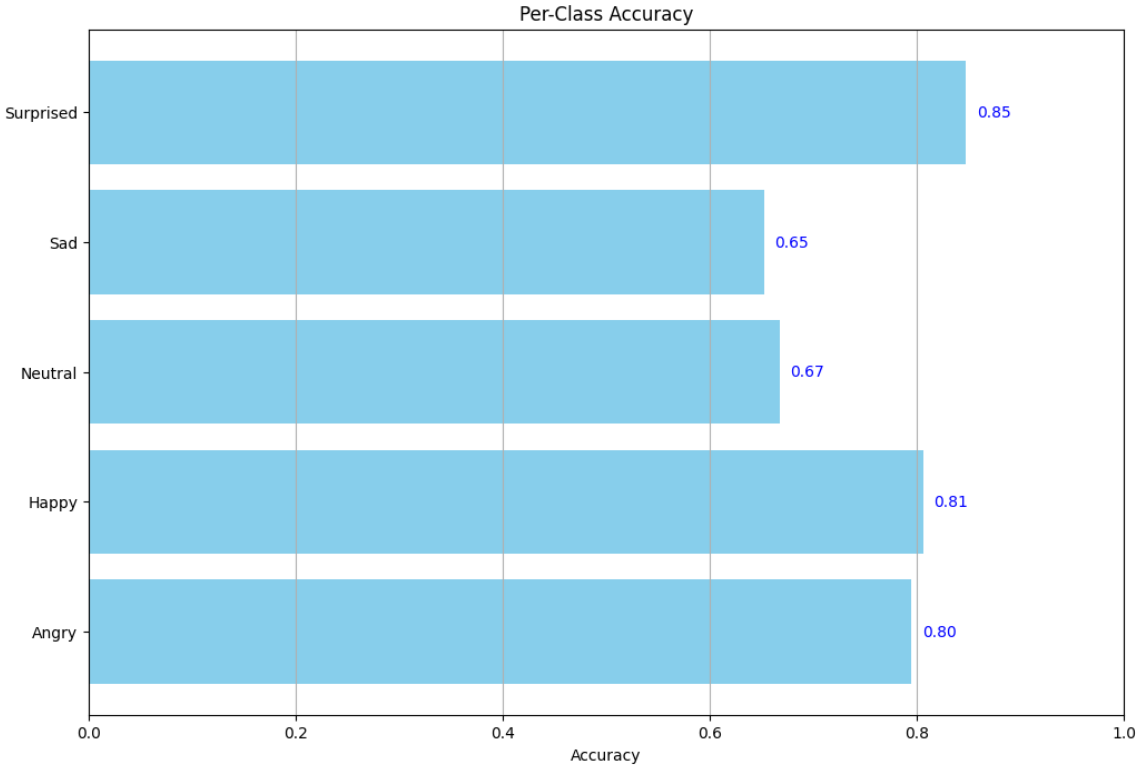

Figura 36: "*Accuracy*"por clase para VGG11 entrenado sin regularización.

- **Con regularizacion L1 con valor de 0.001.**

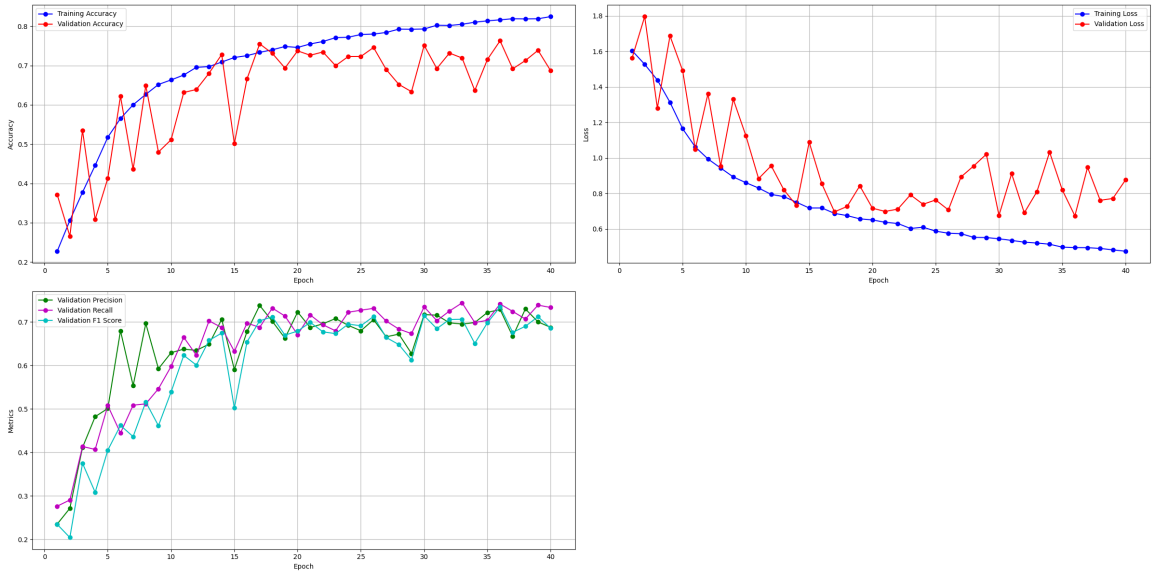

Figura 37: Historial de entrenamiento de VGG11 con regularización L1 con valor de 0.001.

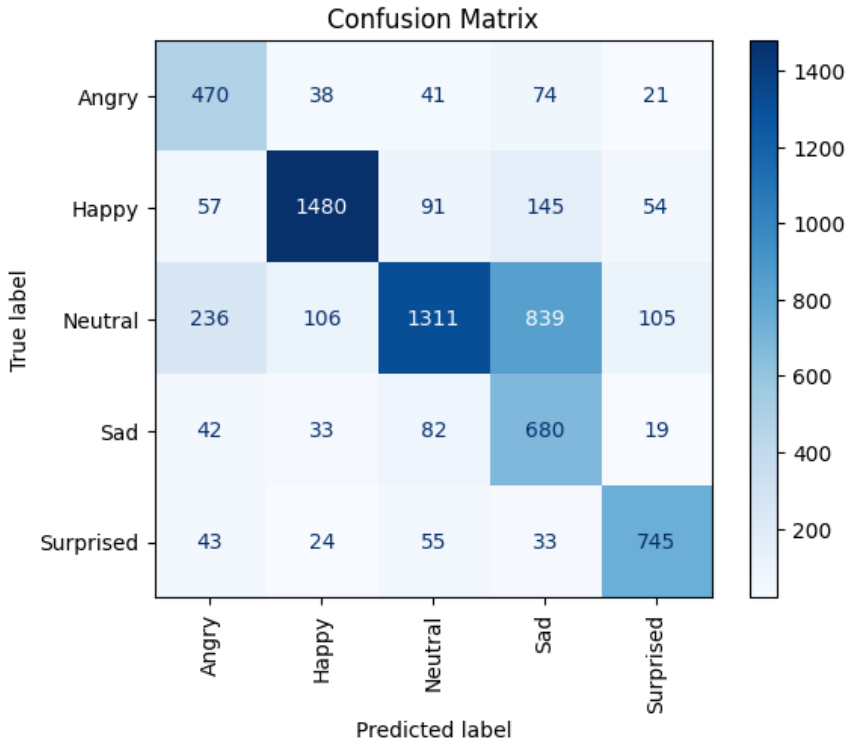

Figura 38: Matriz de confusión para VGG11 entrenadocon regularización L1 con valor de 0.001.

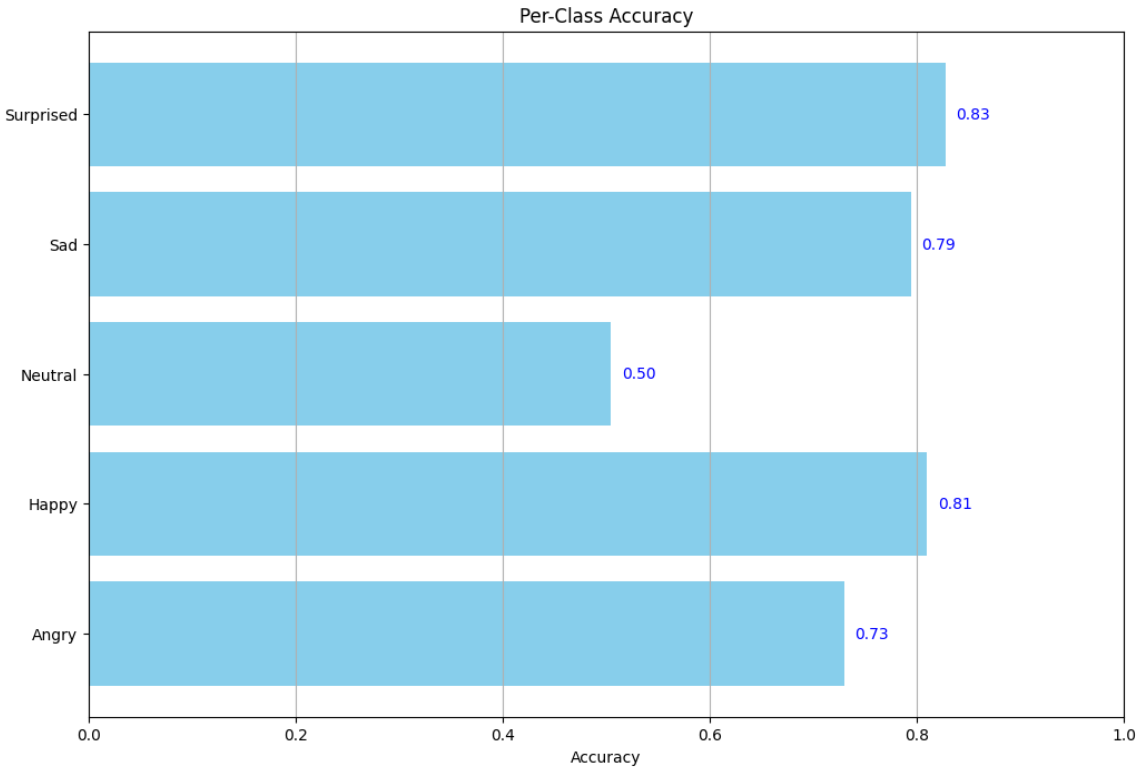

Figura 39: "*Accuracy*"por clase para VGG11 entrenado con regularización L1 con valor de 0.001.

Para esta arquitectura se puede observar una diagonal en la matriz de confusión lo cual indica alta precisión en las predicciones del modelo. Sin embargo, se puede notar un alto número de falsos positivos para la clase neutral, confundiendo las entradas de la clase "*sad*". Por otro lado se obtuvo 0.72 de F1 score.

6.4. VGG Customizada

- **Sin regularización**.

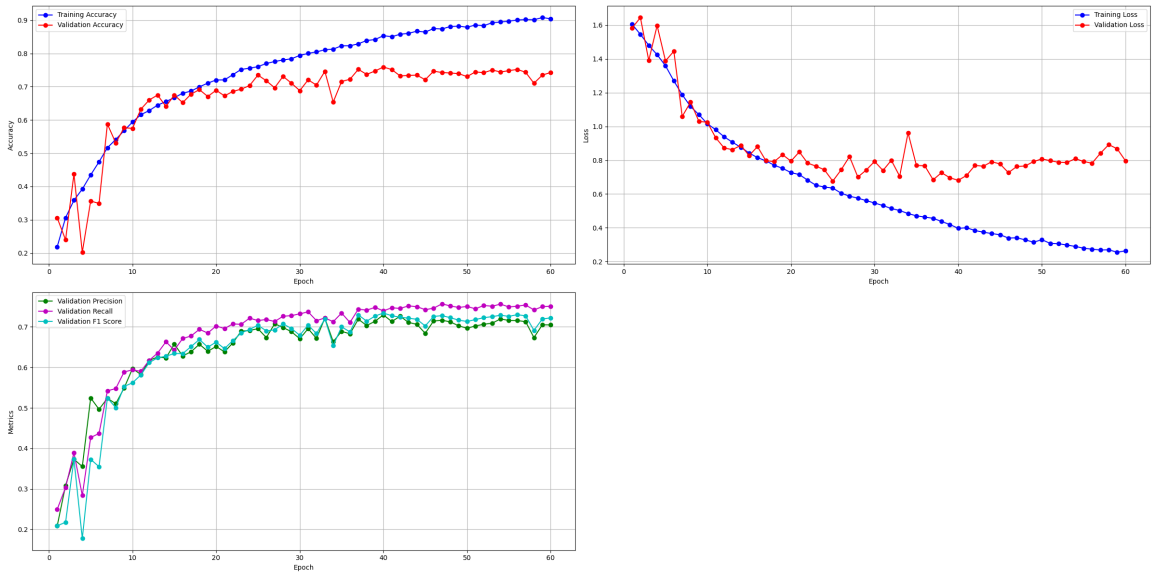

Figura 40: Historial de entrenamiento de VGG customizada sin regularización.

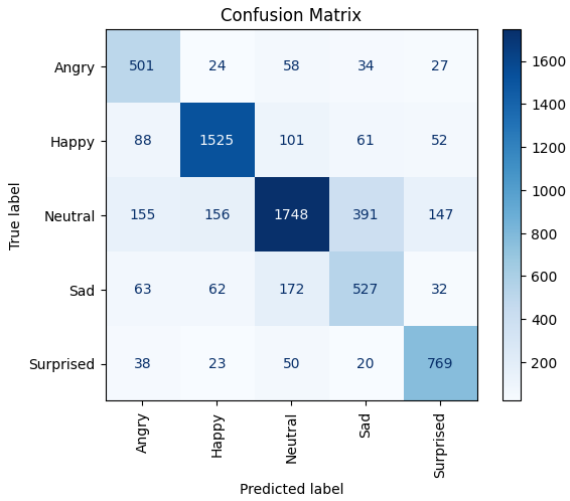

Figura 41: Matriz de confusión para VGG customizada sin regularización..

Arquitectura	Mejor F1 Score
CNN Simple	0.56
ResNet-50	0.65
VGG-11	0.72
VGG Customizada	0.73

Tabla 6.1: Mejor F1 score para cada arquitectura entrenada.

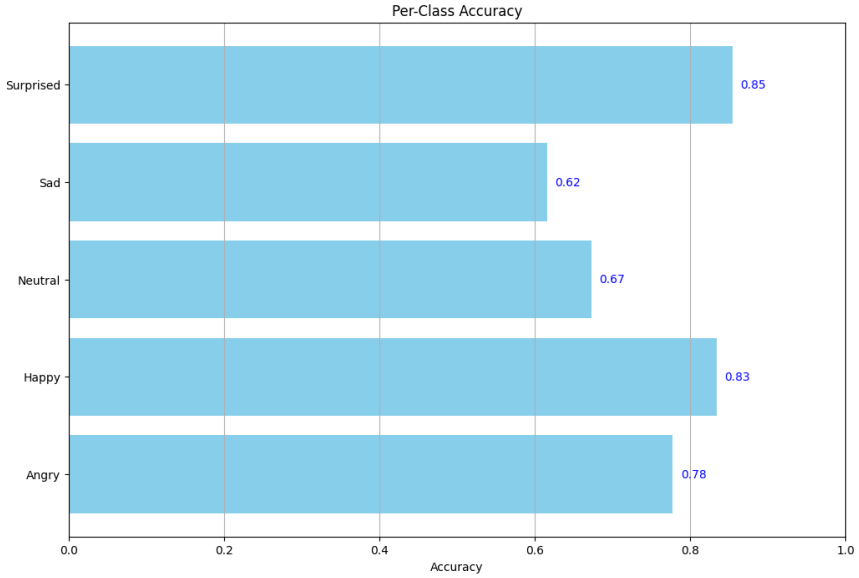

Figura 42: "*Accuracy*"por clase para VGG customizada sin regularización.

Para esta arquitectura se puede observar una diagonal en la matriz de confusión lo cual indica alta precisión en las predicciones del modelo. Además, Se tienen menos falsos negativos de la clase "Neutralz una diagnoal pronunciada en la matriz de confusión. Por otro lado se obtuvo un F1 Score de 0.73.

A partir de los resultados finales, se concluye que fue posible desarrollar un modelo de análisis de sentimientos faciales, para este conjunto de datos, la arquitectura VGG personalizada obtuvo el mejor rendimiento, logrando un F1 score de 0.73.

6.5. Explicabilidad para modelo de clasificación de expresiones faciales

En este caso se analizó el modelo VGG customizada entrenada sin regularización debido a que obtuvo las mejores métricas de rendimiento. A continuación se muestran ejemplos para las 5 clases con explicabilidad de shap.

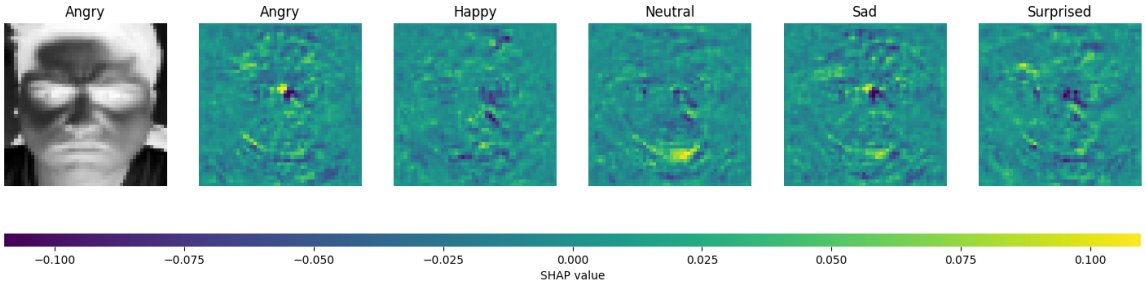

Figura 43: Explicación para una imagen de entrada con valor real "*Angry*". Los píxeles amarillos representan valores SHAP positivos que aumentan la probabilidad de la clase, mientras que los píxeles púrpura representan valores SHAP negativos que reducen la probabilidad de la clase.

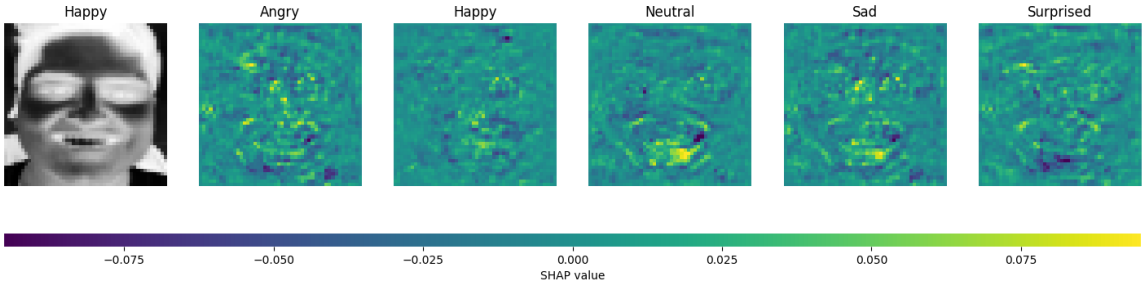

Figura 44: Explicación para una imagen de entrada con valor real "*Happy*". Los píxeles amarillos representan valores SHAP positivos que aumentan la probabilidad de la clase, mientras que los píxeles púrpura representan valores SHAP negativos que reducen la probabilidad de la clase.

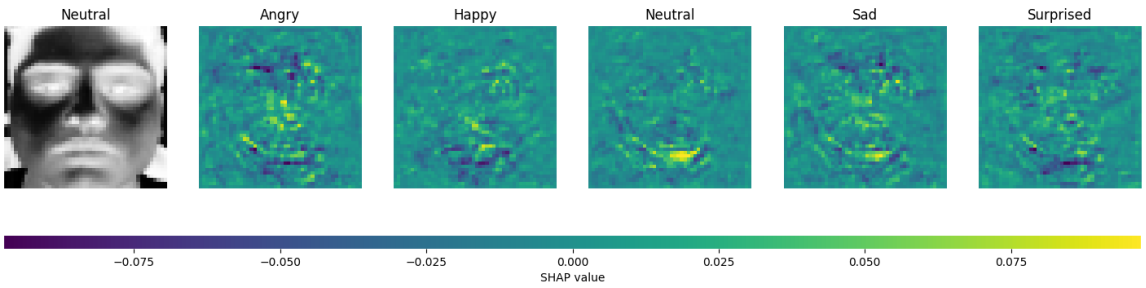

Figura 45: Explicación para una imagen de entrada con valor real "*Neutral*". Los píxeles amarillos representan valores SHAP positivos que aumentan la probabilidad de la clase, mientras que los píxeles púrpura representan valores SHAP negativos que reducen la probabilidad de la clase.

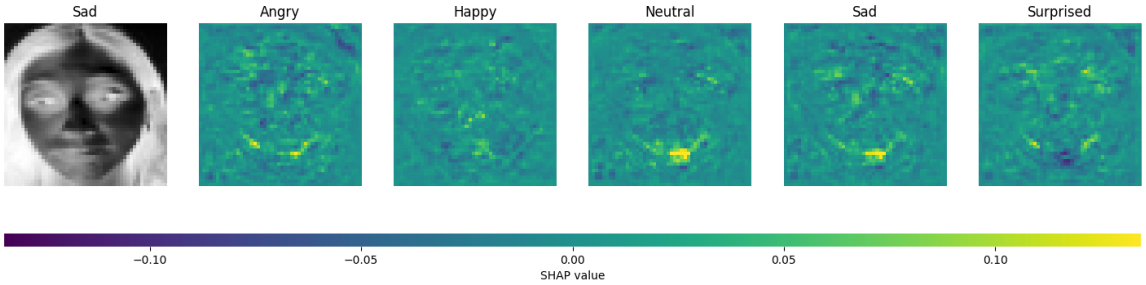

Figura 46: Explicación para una imagen de entrada con valor real "*Sad*". Los píxeles amarillos representan valores SHAP positivos que aumentan la probabilidad de la clase, mientras que los píxeles púrpura representan valores SHAP negativos que reducen la probabilidad de la clase.

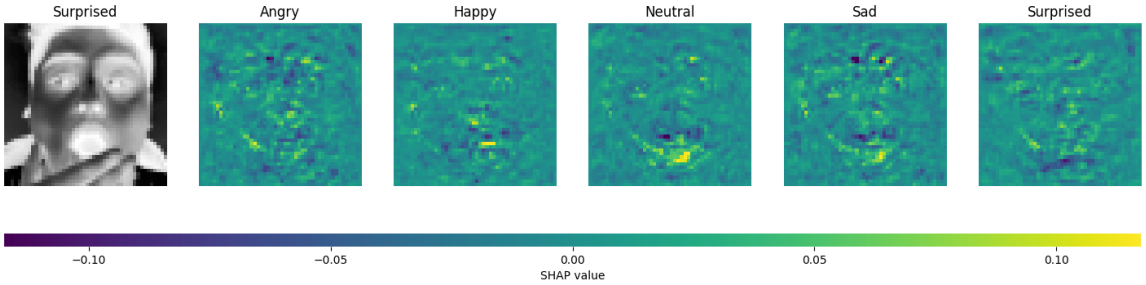

Figura 47: Explicación para una imagen de entrada con valor real "*Surprise*". Los píxeles amarillos representan valores SHAP positivos que aumentan la probabilidad de la clase, mientras que los píxeles púrpura representan valores SHAP negativos que reducen la probabilidad de la clase.

En las figuras anteriores, correspondientes a la explicabilidad del modelo de clasificación de expresiones faciales, se observan áreas de interés específicas en las zonas de la boca y los ojos de los rostros analizados. Esto respalda la validez del modelo, dado que estas regiones son elementos clave en el análisis de expresiones faciales.

6.6. Prueba de Pipeline de reconocimiento de expresiones faciales en tiempo real

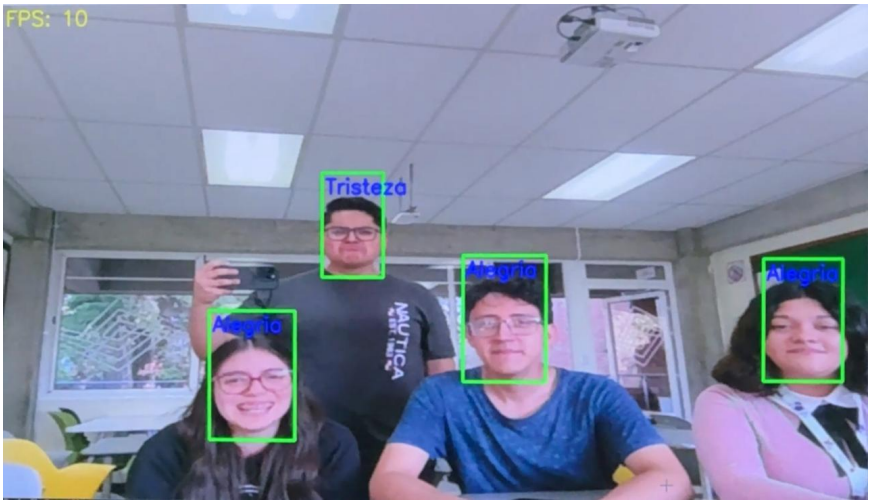

Figura 48: Prueba de de pipeline de reconocimiento de expresiones faciales con 4 participantes.

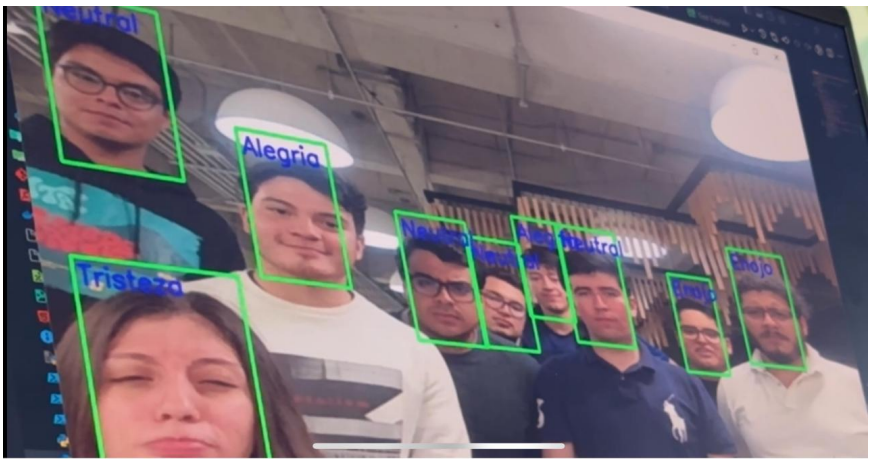

Figura 49: Prueba de de pipeline de reconocimiento de expresiones faciales con 9 participantes.

Esta fase de validación permitió confirmar que el pipeline de reconocimiento de expresiones faciales era lo suficientemente robusto y eficiente para su aplicación en estudios de mercado y análisis en grupos focales de gran escala. Los resultados indicaron que el sistema podía procesar varias fuentes de datos en simultáneo, sin pérdida de rendimiento, lo cual representa un avance significativo en el desarrollo de herramientas de análisis en tiempo real para el estudio de percepciones y reacciones de los participantes.

En estas pruebas se obtuvo una disminución máxima de 6 fotogramas por segundo

cuando se alcanzaba la carga máxima de rostros procesados en paralelo. Cabe mencionar que a pesar de la disminución de fotogramas por segundo, fue posible monitorizar y llevar un seguimiento de las expresiones faciales de los participantes.

6.7. Explicabilidad para modelo de análisis de sentimiento en texto

A continuación se presentan ejemplos del análisis SHAP realizado para el modelo de análisis de sentimiento en texto.

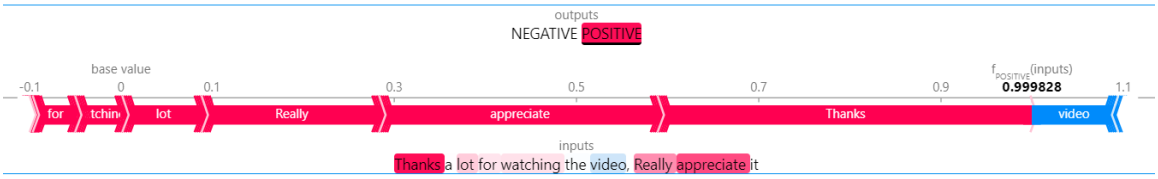

Figura 50: Análisis SHAP para clasificación de sentimiento para texto de entrada con valor real positivo. Las palabras en magenta aumentan la probabilidad de que el modelo clasifique el texto como positivo, mientras que las palabras en azul reducen esta probabilidad.

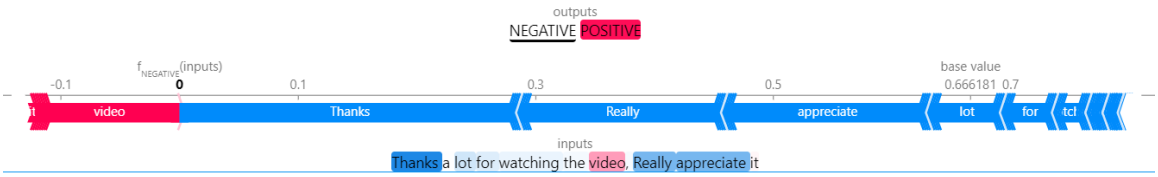

Figura 51: Análisis SHAP para clasificación de sentimiento para texto de entrada con valor real positivo. Las palabras en magenta aumentan la probabilidad de que el modelo clasifique el texto como negativo, mientras que las palabras en azul reducen esta probabilidad.

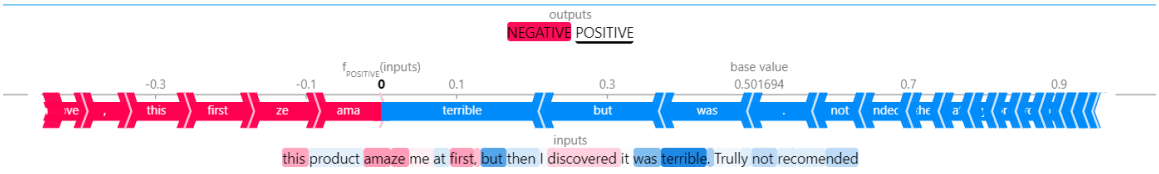

Figura 52: Análisis SHAP para clasificación de sentimiento para texto de entrada con valor real negativo.. Las palabras en magenta aumentan la probabilidad de que el modelo clasifique el texto como positivo, mientras que las palabras en azul reducen esta probabilidad.

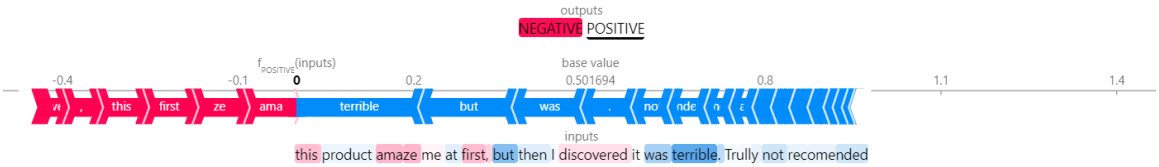

Figura 53: Análisis SHAP para clasificación de sentimiento para texto de entrada con valor real negativo. Las palabras en magenta aumentan la probabilidad de que el modelo clasifique el texto como negativo, mientras que las palabras en azul reducen esta probabilidad.

En este caso, se puede observar la importancia de cada palabra en la clasificación de texto como positivo o negativo utilizando el análisis de SHAP. La intensidad del color refleja la magnitud de la influencia de cada palabra: cuanto más intenso es el color, mayor es el impacto en la predicción final.

CAPÍTULO 7

Análisis de Resulados

En las pruebas realizadas para el pipelne de detección de expresiones faciales, se puede observar que se logran identificar el 100 % de los rostros presentes en la imágen. Detectando 4 rostros en la prueba con 4 participantes (como se muestra en la figura 48) y 9 rostros en la prueba con 9 participantes (como se muestra en la figura 49). Por tanto se puede conlcuir que fue posible desarrollar un detector de rostros basado en filtros de haar y detectores en cascada. Cabe mencionar que en las pruebas realizadas en ocasiones no se detectaron los rostros al estar cerca de una fuente de luz (lampara, luz solar, etc.). Esto puede tener explicación en el funcionamiento de los filtros de haar al momento de realizar la detección, ya que estos se basan en los cambios de contraste para determinar si el objeto objetivo se encuentra presente en la imagen. Al estar cerca de una fuente de luz, el objeto pierde parte de sus cambios de contraste característicos lo cual dificulta su detección. En base a esta observación se recomienda prestar especial atención a las fuentes de iluminación al utilizar un modelo de detección de rostros basado en detectores en cascada y filtros de Haar, ya que estos algoritmos son sensibles a las variaciones de luz.

Para el entrenamiento de la red Convolucional simple (la primera de las arquitecturas con las que se trabajó) se puede notar una tendencia similar al en el "*accuracy*"de entrenamiento como en el de validación. Este comportamiento se repite para la pérdida, precisión, "*Recallz* el "F1 *Score*", como se puede observar en las figuras 19, 22, 25. Sin embargo en las gráficas de "*accuracy*"por clase (figuras 21, 24, y 27), se puede observar que se logra un mejor "*accuracy*"para las clases "*surprisedz* "*happy*". Este comportamiento se reduce al agregar regularización L1, sin embargo no se elimina por completo.

En cuanto a los entrenamientos realizados para la arquitecture ResNet-50, se puede

observar una tendencia al sobreajuste en las figuras 28 y 31 ya que a partir de un punto la tendencia de el "*accuracy*"de entrenamiento y validación divergen, al igual que la pérdida. Por otro lado, al igual que con la red convolucional simple, se puede observar que e logra un mejor "*accuracy*"para las clases "*surprised*z "*happy*"según las figuras 30 y 33. Además, se puede notar un "*accuracy*"basante para la clase "*sad*". En este caso el sobreajuste puede explicarse por la profundidad de la red ResNet-50 ya que cuenta con 50 bloques residuales. Esto la hace propensa a aprender variaciones pequeñas en los datos de entrenamiento que no siempre se ajustan a datos externos. Cabe mencionar que en el caso de esta arquitectura si fue efectiva la regularización L1 para reducir el sobreajuste, reforzando la idea de que este se debe a la profundidad de la misma.

Para los entrenamientos realizados para la arquitectura VGG11, se puede observar una tendencia al sobreajuste, aunque menor que el presentado con ResNet-50 (como se puede observar en las graficas 37 y 34). Sin embargo esta arquitectura presenta una distribución de "*accuracy*za que para el entrenamiento con regularización presenta "*accuracy*"mayor a 0.8 para las clases "*surprised*", "*happy*z "*angy*". Además presenta "*accuracy*"mayor a 0.6 para las clases restantes "*neutral*z "*sad*", como se oobserva en la figura 36. Por otro lado, se puede observar una diagonal pronunciada en la matriz de confusión se pueden notar varios falsos positivos para la clase "*neutral*çonfundiendola especialmente con la clase "*sad*".

Para la arquitectura VGG customizada presenta un sobreajuste como se observa en la figura 40. Sin embargo, al observal la matriz de confusión (figura 41), se puede observar una diagonal pronunciada y menor número de falsos positivos para la clase "*neutral*". Además, se observa una distribución de "*accuracy*"similar a lo obtenido con la arquitectura VGG11. En este caso se tiene mejor "*accuracy*"para las clases "*happy*", "*surprised*z "*angry*"que para "*sad*z "*neutral*", como se observa en la figura 41. Este contraste entre la distribución de "*accuracy*z la matriz de confusión valida la decisión de agregar 2 capas de dropout luego de cada capa "*fully connected*.[en] la arquitectura ya que se logró reducir el sobreajuste.

Al momento de analizar los análisis SHAP realizados para el modelo de clasificación de expresiones faciales se pueden notar que, especialmente para las clases "*neutral*z "*sad*.[el] modelo se concentra en detectar la expresión de la boca ya que para estas regiones se obtuvieron valores shap elevados al analizar imágenes con valor real neutral y "*sad*"(figuras 46 y 47). Por otro lado, para la imagen analizada con valor real "*angry*"(figura **??**) se pueden observar valores shap elevados en el área dónde existe un ceño fruncido en la imagen original. Esto indica que el modelo busca un ceño fruncido como característica para predecir el enojo.

Además, en el ejemplo con valor real "*sad*"(figura 47) se pueden observar valores shap bajos en la parte central de la boca para la clase "*surprised*". Esto indica que para la clase "*surprised*.[el] modelo utiliza la característica de una boca cerrada y con comesuras bajas para descartar la clase "*surprised*". Por otro lado, se pueden observar valores altos para la clase "*sad*.[en] la zona de las cejas. Cabe mencionar que en el ejemplo se observa una persona con los ojos entrecerrados y las cejas un poco

elevadas. En este caso se puede concluir que el modelo utiliza la característica de ojos entrecerrados y cejas altas para detectar tristeza.

En el ejemplo con valor real neutral (figura 46) se pueden observar valores bajos para la clase "*angry*.[en] la zona de las cejas. Esto refueza la ide de que el modelo busca cejas fruncidas para predecir enojo.

Estas características descritas por la explicabilidad pueden verse reflejadas por el análisis shap pueden verse reflejadas en las pruebas realizadas con el pipeline de detección de expresiones faciales en tiempo real. Por ejemplo, en la figura 48 se puede observar que para el sujeto con playera gris y posición más superior en la escena se detecta la expresión de tristeza al presentar ojos entrecerrados y comisuras bajas, características detectadas en el análisis de SHAP. Asimismo, se detecta alegría para la persona más hacia la izquierda la cual presenta una sonrisa. Por otro lado, en la figura 49 se puede apreciar la detección de enojo para el sujeto más hacia la derecha el cual presenta un ceño fruncido. nuevamente característica detectadas en el análisis SHAP.

Por lo tanto, se puede concluir que fue posible realizar un pipeline de reconocimiento de expresiones faciales a partir de un detector de cascada utilizando filtros de haar y un modelo de clasificaci'on de expresiones faciales en rostros y desarrollar un modelo de análisis de sentimientos faciales capaz de clasificar las expresiones emocionales en tiempo real a partir de los rostros encontrados por el algoritmo de detección. Además, fue posible explicar el funcionamiento del modelo de clasificación de expresiones faciales utilizando SHAP como técnica de explicabilidad.

En el caso del modelo de análisis de sentimiento en texto se puede observar que en la figura 50 la cual tiene como valore real el sentimiento positivo, se predice exitosamente el sentimiento. Además, se puede observar que las palabras que más influyen a la predicción "*thanksz* "*appreciate*", ambas palabras que expresan agradecimiento en el idioma inglés. Asimismo en la figura 52 la cual tiene como valor real la emoción negativa, se puede observar que las palabras que más influyen en la predicción son "*terriblez* "*but*". En este caso "*terrible*"si representa un sentimiento negativo y but indica negatividad debido al contexto de la frase que se utilizó como entrada. Ya que en este contexto, la palabra "*but*.[es] la palabra que una la primera parte de la frase con sentimiento positivo y luego explica que ese sentimiento se experimento en el pasado. En ambos casos el modelo fue capaz de predecir exitosamente el sentimiento del texto y se pudieron identificar las palabras más influyentes en la predicción con un análisis SHAP, haciendo concordancia con su significado en el contexto del texto de entrada para cada caso. En base a estas observaciones se puede concluir que fue posible explicar el funcionamiento del modelo de análisis de sentimiento en texto utilizando SHAP como técnica de explicabilidad.

CAPÍTULO 8

Conclusiones

- Se logró desarrollar un algoritmo de detección de rostros utilizando filtros de Haar y el método de cascada, que demostró ser efectivo para identificar y recortar los rostros en videos en tiempo real. El algoritmo fue capaz de procesar múltiples rostros simultáneamente, manteniendo una alta precisión y velocidad en la detección.
- Fue posible el desarrollo de un modelo de análisis de sentimientos faciales que clasifica las expresiones emocionales en tiempo real a partir de los rostros detectados por el algoritmo de detección. El modelo logró una clasificación precisa y en tiempo real, lo que valida su efectividad y demuestra que cumple con el objetivo de proporcionar un análisis emocional confiable y rápido de las expresiones faciales de los participantes.
- Se desarrolló un modelo de análisis de sentimientos en texto, basado en el modelo preesntrenado "*distilbert/distilbert-base-uncased-finetuned-sst-2-english*çapaz de clasificar emociones en distintos fragmentos de texto. Este modelo demostró una alta precisión en la identificación de sentimientos y emociones, siendo capaz de interpretar tanto comentarios positivos como negativos de forma efectiva.
- Fue posible analizar el modelo de clasificación de expresiones faciales y el de análisis de sentimiento en textoutilizando SHAP como técnica de Explainable AI, lo cual permitió interpretar y comprender el impacto de las características clave en sus decisiones.
- Se elaboró una propuesta de pipeline que integra la detección de expresiones faciales y el de análisis de sentimiento en texto, aplicándolos de forma conjunta, lo cual permite analizar en tiempo real tanto las expresiones faciales como los comentarios textuales de los participantes, ofreciendo una visión integral de sus emociones.

CAPÍTULO 9

Recomendaciones

- Para futuras investigaciones, se recomienda ampliar la base de datos de clasificación de expresiones faciales, poniendo especial énfasis en aumentar las muestras correspondientes al sentimiento de miedo. Esto permitiría al modelo identificar de manera precisa cinco emociones primarias (alegría, tristeza, enojo, sorpresa y miedo), brindando una comprensión más completa y equilibrada de las emociones humanas en diversas situaciones. Además, el enriquecimiento de la base de datos, especialmente en categorías con menor representación, ayudaría a reducir el problema de sobreajuste observado en modelos profundos, al permitirles aprender de una mayor variedad de muestras. Esto no solo mejoraría la generalización del modelo, sino que también incrementaría su capacidad de reconocer emociones en diferentes contextos y grupos demográficos.
- Se recomienda la creación de un conjunto de datos de texto específico que capture el contexto de Guatemala, con el fin de desarrollar un modelo de análisis de sentimiento capaz de interpretar adecuadamente el dialecto y expresiones locales. Esto permitiría al modelo comprender y clasificar de manera más precisa los sentimientos expresados por los usuarios guatemaltecos.
- Se recomienda prestar especial atención a las fuentes de iluminación al utilizar un modelo de detección de rostros basado en detectores en cascada y filtros de Haar, ya que estos algoritmos son sensibles a las variaciones de luz. Cambios en el contraste pueden afectar significativamente la precisión del modelo, por lo que es aconsejable asegurar condiciones de iluminación controladas para obtener resultados óptimos.

Bibliografía

[1] Alzubi, J. A. *y cols.*: *An Overview of Machine Learning: Applications and Challenges*. En *Proceedings of the Second National Conference on Computational Intelligence (NCCI 2018)*, volumen 1142 de *IOP Conf. Series: Journal of Physics: Conf. Series*, página 012012. IOP Publishing, 2018.

[2] Anonymous: *5 Barriers to AI Adoption and How Marketers Can Overcome Them*. `https://www.invoca.com/blog/5-barriers-ai-adoption-how-marketers-can-overcome-them`. Accessed: 2024-10-08.

[3] Anonymous: *What is AI Marketing & How Can You Utilise It?* `https://www.salesforce.com/in/resources/guides/role-of-ai-in-marketing/`. Accessed: 2024-10-08.

[4] Anta, Antonio Fernández, Luis Núñez Chiroque, Philippe Morere y Agustín Santos: *Sentiment Analysis and Topic Detection of Spanish Tweets: A Comparative Study of NLP Techniques*. Procesamiento del Lenguaje Natural, 50:45–52, 2013, ISSN 1135-5948. Received 28-11-12, Revised 13-02-13, Accepted 19-02-13.

[5] Barsoum, Emad, Cha Zhang, Cristian Canton Ferrer y Zhengyou Zhang: *Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution*. En *ACM International Conference on Multimodal Interaction (ICMI)*, 2016.

[6] contributors, OpenCV: *OpenCV: Open Source Computer Vision Library*, 2023. `https://github.com/opencv/opencv`, Accessed: 2023-10-25.

[7] Damadi, Saeed, Golnaz Moharrer, Mostafa Cham y Jinglai Shen: *The Backpropagation Algorithm for a Math Student*. En *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Baltimore, MD, 2023. University of Maryland, Baltimore County (UMBC).

[8] Dharmaraj: *Zero-Padding in Convolutional Neural Networks - Dharmaraj - Medium*, Enero 2022. `https://medium.com/@draj0718/zero-padding-in-convolutional-neural-networks-bf1410438e99`.

[9] Dwith, C.: *Evolution of Convolutional Neural Network Architectures and Their Impact on Edge AI.* FEEDFORWARD, 2(3):1–11, July–September 2023. https://github.com/cyndwith/Evolution-of-Convolutional-Neural-Network-CNN-Compute-vs-Memory-bandwidth-for-

[10] F., J., J. F. y J. F.: *10 Amazing Statistics From The World of AI Marketing.* The Marketing Hustle - Marketing & Brand Strategy Insights To Grow Your Business, January 2024. https://themarketinghustle.com/ai-marketing/10-amazing-statistics-from-the-world-of-ai-marketing/#elementor-toc__heading-anchor-1.

[11] He, Kaiming, Xiangyu Zhang, Shaoqing Ren y Jian Sun: *Deep Residual Learning for Image Recognition.* En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 770–778. IEEE, 2016.

[12] Howley, Karen: *The Evolution of AI in Marketing: From Big Data to On-Demand Creative.* KAREN HOWLEY, March 2024. https://www.karenhowley.com/blog/the-evolution-of-ai-in-marketing-from-big-data-to-on-demand-creative.

[13] Jain, Abhishek: *Pooling and their types in CNN - Abhishek Jain - Medium*, Febrero 2024. https://medium.com/@abhishekjainindore24/pooling-and-their-types-in-cnn-4a4b8a7a4611.

[14] LeCun, Yann, Yoshua Bengio y Geoffrey Hinton: *Deep Learning.* Nature, 521(7553):436–444, 2015. https://hal.archives-ouvertes.fr/hal-04206682.

[15] Lundberg, Scott M y Su In Lee: *A Unified Approach to Interpreting Model Predictions.* En Guyon, I., U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan y R. Garnett (editores): *Advances in Neural Information Processing Systems 30*, páginas 4765–4774. Curran Associates, Inc., 2017. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[16] Navarro, Sandra: *Tipos de capas de red neuronal convolucional*, Abril 2024. https://keepcoding.io/blog/tipos-capas-red-neuronal-convolucional/.

[17] O'Shea, Keiron y Ryan Nash: *An Introduction to Convolutional Neural Networks.* En *Proceedings of the 2015 International Conference on Machine Learning*, Ceredigion, SY23 3DB, UK, 2015. Aberystwyth University.

[18] Shah, Snehal, Akshata Bhat, Sumitra Singh, Arya Chavan y Aryan Singh: *Sentiment Analysis.* International Journal of Progressive Research in Engineering Management and Science (IJPREMS), 4(4):1542–1547, 2024.

[19] Simonyan, Karen y Andrew Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recognition.* En *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. http://www.robots.ox.ac.uk/~vgg/research/very_deep/.

[20] Team, Svitla: *Math Behind CNNs for Image Processing | Svitla Systems*, Agosto 2024. `https://svitla.com/blog/math-at-the-heart-of-cnn/`.

[21] Viola, Paul y Michael Jones: *Rapid Object Detection using a Boosted Cascade of Simple Features*. En *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 511–518. IEEE, 2001.