UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Evaluación de la Interacción Humano-Robot Basada en Expresiones Faciales Detectadas por CNN en Entornos Sociales

TESIS

Que para obtener el título de
Ingeniero Mecatrónico

PRESENTAN
Guerrero Martínez Erick
Salgado Miranda Maya Jily

DIRECTOR DE TESIS
Ing. Elorza López Neftalí

Ciudad Universitaria, Cd. Mx., 2026

Agradecimientos

Guerrero Martínez Erick

Al llegar al final de este camino, me resulta difícil encontrar las palabras exactas para expresar todo lo que siento. La carrera de Ingeniería Mecatrónica en la UNAM ha sido una de las etapas más desafiantes, formativas y significativas de mi vida, y sé que todo lo que viví durante estos años no se puede reflejar por completo en unas cuantas líneas. Sin embargo, quiero intentar plasmar aquí mi más sincera gratitud hacia todas las personas que hicieron posible que hoy pueda estar cerrando este capítulo.

A mi papá, Sergio Guerrero, le agradezco por ser mi ejemplo de trabajo, esfuerzo y dedicación. Gracias por enseñarme con tu ejemplo que las cosas que valen la pena requieren sacrificio y constancia. Por cada madrugada, por cada esfuerzo silencioso que hiciste para que yo pudiera tener las oportunidades que tuve, y por siempre impulsarme a dar lo mejor de mí. Tu fortaleza fue mi referencia en los momentos más difíciles de la carrera, y gran parte de lo que soy como persona y como profesionista es gracias a ti.

A mi mamá, Adelaida Martínez, le agradezco desde lo más profundo de mi corazón por su amor incondicional, su paciencia infinita y su apoyo constante. Gracias por cada palabra de aliento cuando sentía que no podía más, por tu confianza en mí incluso cuando yo mismo dudaba, y por nunca soltar mi mano a lo largo de este proceso. Tu cariño, tu dedicación y tu fuerza fueron el motor que me impulsó a seguir adelante día tras día. No existe forma de agradecerte lo suficiente por todo lo que has hecho por mí.

A mi hermano, Sergio Guerrero, le agradezco por su compañía, su apoyo y por ser parte esencial de mi vida. Gracias por estar ahí en los buenos y en los malos momentos, por tu comprensión en los días de estrés y desvelo, y por recordarme, con tu presencia, la importancia de la familia. Saber que cuento contigo me dio tranquilidad en los momentos de mayor presión.

A mi familia en general, a cada persona que forma parte de mi entorno, les agradezco por su apoyo desinteresado y por creer en mí. Muchos de ustedes estuvieron presentes en momentos donde el camino se veía complicado, donde la carga parecía demasiada y donde la meta se sentía lejana. En esos momentos, sin pedir nada a cambio, extendieron su mano, me ofrecieron palabras de aliento y me dijeron "ya queda menos, échale ganas". Esas palabras, que en su momento parecían simples, fueron las que me dieron la fuerza para no rendirme.

A mi novia y compañera de tesis, Maya Jily Salgado Miranda, le agradezco por haber compartido conmigo este proceso tan importante. Gracias por tu amor, tu paciencia, tu esfuerzo y tu compromiso durante la realización de este trabajo. Recorrer este camino juntos, con sus dificultades y satisfacciones, hizo que los momentos difíciles fueran más llevaderos y

que los logros fueran más significativos. Tu compañía, tu dedicación y tu apoyo incondicional fueron fundamentales para que esta tesis llegara a su conclusión.

A nuestro asesor de tesis, Ing. Elorza López Neftalí, le agradezco profundamente por abrirnos las puertas para poder desarrollar este proyecto. Gracias por su orientación, por su tiempo, por proporcionarnos el material necesario y por apoyarnos en cada etapa del proceso. Pero sobre todo, le agradezco por ayudarnos en un tema tan interesante y motivador, que despertó en nosotros la curiosidad por investigar, experimentar y proponer nuestra propia solución. Su confianza en nuestro trabajo y su guía nos permitieron crecer no solo como estudiantes, sino como ingenieros capaces de enfrentar problemas reales con herramientas reales.

A mis profesores de la Facultad de Ingeniería, les agradezco por cada enseñanza, cada exigencia y cada lección que contribuye a formarme a lo largo de la carrera. Cada uno de ustedes aportó algo valioso a mi formación, y gracias a su dedicación y compromiso con la docencia pude adquirir los conocimientos y las habilidades que hoy me permiten llamarme ingeniero mecatrónico.

Finalmente, quiero reconocer que este logro no es solo mío. Es el resultado del esfuerzo, el amor y la confianza de cada una de las personas que menciono aquí y de muchas otras que, aunque no nombre, saben que forman parte de esta historia. La carrera de Ingeniería Mecatrónica me enseñó que los grandes retos no se enfrentan solos, y esta tesis es la prueba de ello, gracias de corazón.

Salgado Miranda Maya Jily

No hay palabras que logren expresar mi agradecimiento de la manera en que me gustaría; sin embargo, en estas líneas deseo plasmar mis sentimientos más sinceros hacia todas aquellas personas que formaron parte de este camino.

Agradezco profundamente a mi familia, porque nunca me dejó sola. Siempre estuvieron presentes, quizá no siempre físicamente, pero sí a través de su amor, su apoyo, su comprensión y sus palabras de aliento. Gracias por acompañarme en los momentos difíciles, de esfuerzo, cansancio y duda, así como también en los momentos de alegría. Su compañía, confianza y cariño fueron fundamentales para que pudiera llegar hasta aquí.

De manera muy especial, quiero agradecer a mi mamá, Claudia Miranda, por ser mi pilar durante toda esta etapa. Gracias por todo tu esfuerzo, por siempre dar lo mejor de ti por tu familia, por jamás dejarme sola, por nunca soltar mi mano y por estar para mí en todo momento. No hay palabras suficientes para agradecerte todo lo que has hecho y dado por mí sin esperar nada a cambio. Gracias por tu amor incondicional, por tus palabras de aliento: "No te rindas, Mega, tú puedes"; por recordarme siempre que estás orgullosa de mí, que no debo rendirme y que soy capaz de lograr todo lo que me proponga.

Gracias por comprenderme y brindarme tu apoyo incondicional en los momentos más complicados, por enseñarme a no rendirme, a ser perseverante y a no abandonar mis sueños sin importar las dificultades. Gracias por darme las oportunidades que tengo, por ayudarme a cumplir esta meta y, sobre todo, por creer en mí incluso en los momentos en los que yo misma dudaba. Este logro también es tuyo, porque detrás de cada paso que di estuvieron tu apoyo, tu cariño y tu confianza en mí. Te amo infinitamente, mamá; eres mi mayor motivación para ser mejor persona día a día.

A mi hermano, Oscar Salgado, le agradezco por su cariño, su compañía y por ser parte fundamental de mi vida. Gracias por los momentos compartidos, tanto los buenos como los difíciles; por tu apoyo silencioso pero constante; por sacarme una sonrisa incluso cuando sentía que me derrumbaba; por escucharme y por motivarme con tu presencia a querer ser una mejor persona. Gracias por esas tardes de risas y juegos que sé que compartías conmigo para ayudarme a despejar mis preocupaciones; por esas tardes de charla y por confiar en mí. Saber que cuento contigo me ha dado fuerza en más ocasiones de las que imaginas.

A mi novio y compañero de tesis, Erick Guerrero Martínez, le agradezco por compartir conmigo esta etapa tan importante y significativa. Gracias por tu amor, tu esfuerzo, tu apoyo incondicional, tu paciencia y tu compromiso durante la realización de esta tesis. Compartir juntos este camino no fue fácil; enfrentamos dificultades, momentos de cansancio y retos importantes, pero al final lo logramos. Gracias por hacer más llevadera esta etapa, por

caminar a mi lado y por compartir conmigo este logro. Tu compañía, dedicación y apoyo fueron fundamentales para que esta tesis llegara a su conclusión.

Agradezco a la Universidad Nacional Autónoma de México, mi alma mater, por abrirme las puertas al conocimiento y brindarme la oportunidad de formarme profesionalmente en una institución de la que me siento profundamente orgullosa. Asimismo, agradezco a la Facultad de Ingeniería por ser el espacio donde crecí académica y personalmente, por los recursos, laboratorios y oportunidades que puso a mi alcance, y por forjar en mí el pensamiento crítico, la disciplina y el compromiso que hoy me definen.

A nuestro asesor, Ing. Elorza López Neftalí, le agradezco profundamente por su orientación, paciencia, tiempo y conocimientos compartidos durante el desarrollo de este trabajo. Gracias por creer en este proyecto desde el inicio, por guiarnos con dedicación y por brindarnos la confianza necesaria para explorar nuevas ideas. Su experiencia y sus observaciones no solo enriquecieron esta investigación, sino que también dejaron en mí enseñanzas valiosas que llevaré conmigo en mi vida profesional.

A mis profesores, les agradezco profundamente por cada enseñanza compartida durante mi formación. Gracias por su dedicación, paciencia y compromiso, por impulsarme a pensar de manera crítica, a cuestionar, a aprender de los errores y a esforzarme por ser mejor. Cada lección, dentro y fuera del aula, contribuyó a construir las bases de este logro y dejó en mí aprendizajes que llevaré conmigo en mi vida profesional y personal.

También quiero dedicar un agradecimiento muy especial a mi perrita, Camila, quien, aunque ya no pudo acompañarme hasta el final de este proceso, dejó una huella inmensa en mi corazón. Durante su vida me brindó amor, compañía y consuelo, estando a mi lado de manera incondicional en muchas noches de desvelo y en momentos difíciles. Su presencia fue una fuente de alegría, y su recuerdo estuvo presente a lo largo de este camino.

Finalmente, reconozco el esfuerzo, la constancia y la fortaleza que me permitieron llegar hasta este momento. Este camino no estuvo libre de dificultades, pero cada reto me dejó aprendizajes valiosos y me ayudó a crecer personal y profesionalmente. Esta tesis representa no solo el cierre de una etapa académica, sino también el resultado de perseverancia, disciplina, compromiso y amor por lo que he construido a lo largo de este proceso.

Este logro no es solo mío; por eso, a todos, gracias de corazón.

Índice

Resumen

El presente trabajo desarrolla un sistema inteligente de reconocimiento automático de expresiones faciales mediante técnicas de aprendizaje automático y aprendizaje profundo, orientado a mejorar la calidad de la interacción humano-robot en entornos sociales. El sistema identifica seis estados emocionales (felicidad, tristeza, enojo, asombro, disgusto y neutralidad) a partir de imágenes faciales capturadas bajo condiciones controladas.

Se implementaron y evaluaron cinco modelos de clasificación con enfoques complementarios. En el enfoque basado en imágenes, se desarrolló una red neuronal convolucional (CNN) base de tres capas convolucionales que procesa imágenes de 96×96 píxeles en escala de grises, y una CNN mejorada basada en Transfer Learning con MobileNetV2, preentrenada sobre ImageNet y ajustada en dos fases con técnicas de aumento de datos. En el enfoque basado en características geométricas, se entrenó un perceptrón multicapa (MLP) alimentado con los 1404 valores correspondientes a los 468 puntos faciales tridimensionales (coordenadas x, y, z) extraídos mediante MediaPipe Face Mesh, una Máquina de Vectores de Soporte (SVM) con kernel RBF, y un modelo de Bosque Aleatorio (Random Forest), ambos alimentados con 11 características geométricas normalizadas extraídas con MediaPipe.

Para el entrenamiento de los modelos, se construyó una base de datos original compuesta por aproximadamente 900 imágenes (150 por emoción), capturadas mediante un protocolo estandarizado en un estudio fotográfico montado específicamente para este proyecto en las instalaciones del Centro de Ingeniería Avanzada (CIA) de la Facultad de Ingeniería, UNAM. El proceso incluyó el uso de equipo profesional de iluminación, eliminación digital de fondos mediante software de edición, y preprocesamiento sistemático de todas las muestras.

Los modelos fueron entrenados y evaluados utilizando TensorFlow, Keras y scikit-learn, empleando técnicas de regularización, normalización y optimización adaptativa. La CNN base alcanzó una precisión de 98.89%, la CNN con MobileNetV2 obtuvo 92.78%, el MLP alcanzó 86.67%, el SVM logró 98.33% y el Random Forest obtuvo 99.44% en el conjunto de prueba. Los altos valores de precisión se atribuyen a las condiciones controladas de captura y al uso de un único sujeto, lo cual valida el sistema como prototipo funcional y prueba de concepto para aplicaciones de reconocimiento emocional.

Este trabajo contribuye al área de robótica social al proporcionar una solución tecnológica funcional que permite a sistemas automatizados identificar estados emocionales humanos, facilitando interacciones más naturales y sensibles al estado emocional del usuario. La evaluación comparativa entre los cinco modelos permite identificar las ventajas y limitaciones de cada arquitectura, facilitando la toma de decisiones informadas para futuros desarrollos en el área de computación afectiva e interacción humano-robot.

Palabras clave: Reconocimiento de emociones, redes neuronales convolucionales, Transfer Learning, MediaPipe, interacción humano-robot, visión artificial, aprendizaje profundo, SVM, Random Forest.

Abstract

This work develops an intelligent system for automatic facial expression recognition using machine learning and deep learning techniques, aimed at improving the quality of human-robot interaction in social environments. The system identifies six emotional states (happiness, sadness, anger, astonishment, disgust, and neutrality) from facial images captured under controlled conditions.

Five classification models with complementary approaches were implemented and evaluated. In the image-based approach, a base Convolutional Neural Network (CNN) with three convolutional layers was developed to process 96×96-pixel grayscale images, along with an improved CNN based on Transfer Learning using MobileNetV2, pretrained on ImageNet and fine-tuned in two phases with data augmentation techniques. In the geometric feature-based approach, a Multilayer Perceptron (MLP) was trained using the 1404 values corresponding to the 468 three-dimensional facial landmarks (x, y, z coordinates) extracted through MediaPipe Face Mesh, a Support Vector Machine (SVM) with RBF kernel, and a Random Forest model, both fed with 11 normalized geometric features extracted via MediaPipe.

For model training, an original database consisting of approximately 900 images (150 per emotion) was constructed, captured through a standardized protocol in a photography studio specifically set up for this project at the Advanced Engineering Center (CIA) facilities of the Faculty of Engineering, UNAM. The process included the use of professional lighting equipment, digital background removal through editing software, and systematic preprocessing of all samples.

The models were trained and evaluated using TensorFlow, Keras, and scikit-learn, employing regularization, normalization, and adaptive optimization techniques. The base CNN achieved an accuracy of 98.89%, the MobileNetV2 CNN obtained 92.78%, the MLP reached 86.67%, the SVM achieved 98.33%, and the Random Forest obtained 99.44% on the test set. The high accuracy values are attributed to the controlled capture conditions and the use of a single subject, which validates the system as a functional prototype and proof of concept for emotion recognition applications.

This work contributes to the field of social robotics by providing a functional technological solution that enables automated systems to identify human emotional states, facilitating more natural and emotionally aware interactions. The comparative evaluation across five models identifies the strengths and limitations of each architecture, supporting informed decision-making for future developments in affective computing and human-robot interaction.

Keywords: Emotion recognition, convolutional neural networks, Transfer Learning, MediaPipe, human-robot interaction, computer vision, deep learning, SVM, Random Forest.

Índice de figuras

Índice de tablas

Lista de acrónimos

Acrónimo	Significado
API	Interfaz de Programación de Aplicaciones *(Application Programming Interface)*
AUs	Unidades de Acción Facial *(Action Units)*
CIA	Centro de Ingeniería Avanzada
CNN	Red Neuronal Convolucional *(Convolutional Neural Network)*
CPU	Unidad Central de Procesamiento *(Central Processing Unit)*
FACS	Sistema de Codificación de Acción Facial *(Facial Action Coding System)*
GB	Gigabyte
GPU	Unidad de Procesamiento Gráfico *(Graphics Processing Unit)*
HCI	Interacción Humano–Computadora *(Human–Computer Interaction)*
HRI	Interacción Humano–Robot *(Human–Robot Interaction)*
IA	Inteligencia Artificial
LSTM	Memoria a Largo y Corto Plazo *(Long Short-Term Memory)*
MLP	Perceptrón Multicapa / Multilayer Perceptron
RAM	Memoria de Acceso Aleatorio *(Random Access Memory)*
RBF	Función de Base Radial / Radial Basis Function
ReLU	Unidad Lineal Rectificada *(Rectified Linear Unit)*
RF	Random Forest
RGB	Rojo, Verde y Azul *(Red, Green, Blue)*
RNN	Red Neuronal Recurrente *(Recurrent Neural Network)*
SGD	Descenso Estocástico del Gradiente *(Stochastic Gradient Descent)*

SVM	Máquina de Vectores de Soporte / Support Vector Machine
TPU	Unidad de Procesamiento Tensorial *(Tensor Processing Unit)*
UNAM	Universidad Nacional Autónoma de México

Capítulo 1. Introducción

1.1 Introducción

La capacidad de reconocer y responder a las emociones humanas constituye un elemento fundamental en cualquier interacción social significativa. Tradicionalmente, esta habilidad ha sido exclusiva de los seres humanos, quienes pueden interpretar señales faciales, corporales y vocales para comprender el estado emocional de sus interlocutores y ajustar su comportamiento en consecuencia. Sin embargo, con el avance de la inteligencia artificial y el creciente despliegue de sistemas autónomos en entornos sociales, surge la necesidad de dotar a las máquinas de capacidades perceptuales similares.

El reconocimiento automático de emociones a partir de expresiones faciales representa uno de los desafíos más relevantes en el campo de la visión por computadora y la interacción humano-computadora. Las expresiones faciales son uno de los canales de comunicación no verbal más ricos y universales, capaces de transmitir información emocional de manera rápida e intuitiva. Investigaciones como las de Paul Ekman han demostrado que ciertas emociones básicas se manifiestan mediante patrones musculares faciales reconocibles a nivel intercultural, lo que abre la posibilidad de desarrollar sistemas computacionales capaces de interpretarlas de manera automática.

En el contexto de la robótica social, la capacidad de un sistema para reconocer emociones humanas es esencial para lograr interacciones más fluidas y empáticas. Robots asistenciales, tutores inteligentes, sistemas de atención al cliente y agentes conversacionales pueden beneficiarse enormemente de esta tecnología, ya que les permite ajustar su comportamiento en función del estado afectivo del usuario, mejorando así la calidad y efectividad de la interacción.

A pesar de los avances recientes en aprendizaje profundo y visión artificial, el reconocimiento de emociones faciales sigue presentando desafíos técnicos significativos. La variabilidad en condiciones de iluminación, ángulos de captura, características individuales de los rostros y sutilezas en las expresiones hacen necesario el desarrollo de modelos robustos capaces de generalizar adecuadamente ante nuevas situaciones.

El presente trabajo aborda este problema mediante el diseño, implementación y evaluación de un sistema completo de reconocimiento emocional basado en cinco modelos de clasificación con enfoques complementarios. En el enfoque basado en imágenes, se desarrolló una red neuronal convolucional (CNN) base que procesa directamente imágenes faciales en escala de grises, y una CNN mejorada mediante Transfer Learning con MobileNetV2 preentrenada sobre ImageNet. En el enfoque basado en características geométricas, se implementaron un perceptrón multicapa (MLP) que analiza 468 puntos faciales extraídos mediante MediaPipe Face Mesh, una Máquina de Vectores de Soporte (SVM) y un modelo de Bosque Aleatorio

(Random Forest), ambos alimentados con características geométricas normalizadas. Esta combinación permite aprovechar tanto la capacidad de las CNN para detectar patrones visuales complejos, como la precisión y eficiencia de los modelos basados en análisis geométrico de puntos faciales.

Una característica distintiva de este proyecto es la construcción de una base de datos original mediante la captura sistemática de imágenes en un estudio fotográfico montado específicamente para tal fin en las instalaciones del Centro de Ingeniería Avanzada (CIA) de la Facultad de Ingeniería, UNAM. Este proceso no solo implicó el desarrollo técnico del sistema de reconocimiento, sino también el aprendizaje y aplicación de técnicas de fotografía profesional, diseño de iluminación, edición digital de imágenes y preprocesamiento de datos.

1.2 Objetivos

Objetivo general:

Diseñar e implementar un sistema de reconocimiento automático de expresiones faciales basado en técnicas de aprendizaje automático y aprendizaje profundo, orientado a mejorar la interacción humano-robot en entornos sociales.

Objetivos específicos:

1. Construir una base de datos de imágenes faciales con seis emociones (felicidad, tristeza, enojo, asombro, disgusto y neutralidad) bajo un protocolo estandarizado de captura.
2. Implementar y evaluar cinco modelos de clasificación emocional con enfoques complementarios: CNN Base, CNN con Transfer Learning (MobileNetV2), MLP, SVM y Random Forest.
3. Realizar un análisis comparativo del rendimiento de los cinco modelos bajo condiciones equivalentes de evaluación y sobre el mismo conjunto de datos.
4. Desarrollar una aplicación de demostración en tiempo real que integre los modelos entrenados para la detección de emociones mediante cámara web.

1.3 Estructura del documento

El trabajo se estructura de la siguiente manera: el Capítulo 2 presenta el marco teórico que sustenta el proyecto, abordando conceptos clave de inteligencia artificial, aprendizaje profundo, visión artificial, robótica social y reconocimiento de emociones; el Capítulo 3 describe la metodología empleada, incluyendo el diseño experimental, las herramientas utilizadas, el proceso de captura y preprocesamiento de imágenes, y la arquitectura de los

modelos implementados; el Capítulo 4 detalla el desarrollo del sistema y la integración de componentes; el Capítulo 5 presenta los resultados obtenidos mediante evaluación cuantitativa y cualitativa; finalmente, el Capítulo 6 expone las conclusiones, limitaciones del trabajo y posibles líneas de investigación futuras.

Este trabajo contribuye al campo de la interacción humano-robot al proporcionar una solución tecnológica funcional, replicable y bien documentada, que puede servir como base para futuros desarrollos en el área de computación afectiva y robótica social.

Capítulo 2. Marco Teórico

2.1 Inteligencia Artificial

La inteligencia artificial (IA) es la rama de la informática dedicada a diseñar algoritmos y sistemas capaces de llevar a cabo tareas que, realizadas por una persona, exigirían habilidades cognitivas como razonar, percibir, aprender, planificar o tomar decisiones. Su meta central es construir agentes inteligentes: sistemas que perciben su entorno, razonan sobre él, aprenden de la experiencia y actúan de forma autónoma para alcanzar objetivos determinados [1].

A lo largo de su historia, la disciplina ha evolucionado en varias direcciones teóricas y prácticas. Russell y Norvig [1] resumen las distintas maneras de entender y construir sistemas inteligentes en cuatro grandes categorías:

- Sistemas que piensan como humanos, inspirados en modelos del procesamiento mental y la cognición.
- Sistemas que actúan como humanos, capaces de comportarse de forma indistinguible de una persona, como ciertos robots sociales o agentes conversacionales.
- Sistemas que piensan racionalmente, fundados en la lógica formal y la teoría de la decisión.
- Sistemas que actúan racionalmente, conocidos como agentes racionales, que eligen la acción que maximiza el resultado esperado según su entorno.

Otra clasificación habitual atiende al alcance de sus capacidades [21]:

- IA débil o estrecha, especializada en una tarea concreta —un sistema de recomendación o un detector de rostros, por ejemplo—. Es la que existe hoy en la práctica.
- IA general, que igualaría la versatilidad cognitiva humana y todavía se encuentra en fase de investigación.
- IA superinteligente, un escenario teórico en el que un sistema superaría a los humanos en prácticamente todos los dominios.

El notable avance de la IA en las últimas décadas no responde a un solo factor, sino a la coincidencia de tres: el crecimiento del poder de cómputo, el acceso a grandes volúmenes de datos (big data) y el desarrollo de algoritmos de aprendizaje más eficientes. Gracias a esa combinación, problemas antes intratables —el reconocimiento del habla, la traducción automática, la conducción autónoma o la visión por computadora— hoy se resuelven

mediante redes neuronales y aprendizaje profundo [1]. Como disciplina, la IA se apoya además en la lógica, la estadística, la teoría de la información y las ciencias cognitivas, y admite dos enfoques complementarios: el simbólico, basado en reglas y razonamiento explícito, y el conexionista, basado en redes que aprenden a partir de ejemplos [21].

Este trabajo se inscribe de lleno en el enfoque conexionista. Aquí la IA es el componente que transforma la imagen de un rostro en una emoción estimada: los modelos aprenden, a partir de ejemplos etiquetados, a asociar la configuración del rostro con un estado afectivo. Esa estimación es la que después permite que el sistema ajuste su respuesta, y constituye el primer eslabón de la interacción humano-robot que el proyecto busca habilitar.

2.2 Machine Learning

El aprendizaje automático (Machine Learning) es la subdisciplina de la inteligencia artificial que estudia cómo lograr que un programa aprenda a partir de datos, en lugar de ser programado de forma explícita para cada situación. La idea central es sencilla pero poderosa: en vez de escribir reglas fijas para resolver un problema, se le presentan al modelo numerosos ejemplos y se deja que ajuste por sí mismo sus parámetros internos durante un proceso de entrenamiento, mejorando su desempeño conforme acumula experiencia. Ese ajuste guiado por los datos es, precisamente, lo que separa al aprendizaje automático de la programación tradicional.

Según la naturaleza de los datos y el tipo de retroalimentación disponible, suelen distinguirse tres grandes paradigmas:

- Aprendizaje supervisado: el modelo aprende de datos etiquetados, es decir, ejemplos en los que se conoce la respuesta correcta —por ejemplo, imágenes acompañadas de la emoción que representan—. Su objetivo es generalizar esa relación para predecir la etiqueta de datos nuevos.
- Aprendizaje no supervisado: trabaja con datos sin etiqueta y busca descubrir en ellos estructuras ocultas, como agrupamientos o patrones de similitud.
- Aprendizaje por refuerzo: un agente aprende a decidir mediante prueba y error, guiado por las recompensas y penalizaciones que recibe al interactuar con su entorno.

Este proyecto se enmarca en el primero de esos paradigmas. Se trabaja con aprendizaje supervisado: cada imagen facial de la base de datos se etiqueta con la emoción correspondiente y, a partir de ese conjunto, se entrenan los distintos modelos que después clasifican el estado afectivo de una persona a partir de una imagen no vista [9], [18], [19].

2.3 Deep Learning

El aprendizaje profundo (Deep Learning) es un subcampo del aprendizaje automático que emplea redes neuronales artificiales de múltiples capas —las llamadas redes profundas— para construir, capa tras capa, representaciones cada vez más abstractas de los datos. Cada capa toma la salida de la anterior y la transforma: las primeras detectan rasgos simples y las últimas, conceptos complejos. En una imagen facial, por ejemplo, las capas iniciales responden a bordes y texturas, mientras que las profundas capturan configuraciones completas del rostro.

La diferencia esencial con el aprendizaje automático clásico está en cómo se obtienen las características. En los métodos tradicionales, un experto debe diseñar a mano los atributos que describen cada dato, una tarea costosa y dependiente del dominio. Una red profunda, en cambio, los aprende por sí misma directamente de los datos crudos —imágenes, texto o audio— durante el entrenamiento. Esa capacidad de descubrir de forma automática las características relevantes es la que la ha vuelto tan efectiva en problemas de percepción.

Entre las tareas donde ha mostrado un desempeño sobresaliente destacan:

- El reconocimiento de imágenes y objetos.
- La traducción automática entre idiomas.
- El reconocimiento de voz.
- La clasificación de emociones faciales, como la que aborda este proyecto.

Su éxito, no obstante, no depende solo de la arquitectura: requiere grandes volúmenes de datos, hardware capaz de procesarlos (GPU y TPU) y algoritmos de entrenamiento eficientes, como la retropropagación del error junto con optimizadores modernos —Adam o SGD, entre otros— que ajustan los parámetros de la red de forma estable [8].

2.4 Transfer Learning

El Transfer Learning, o aprendizaje por transferencia, es una técnica en la que el conocimiento que un modelo adquirió al resolver una tarea se reutiliza como punto de partida para abordar otra distinta pero relacionada [24]. La diferencia con el entrenamiento convencional es notable: mientras que allí los pesos de la red se inicializan al azar y todo se aprende desde cero, aquí se parte de representaciones que el modelo ya domina, lo que recorta de manera considerable la cantidad de datos y el tiempo de cómputo necesarios para alcanzar un buen desempeño [24].

Esta ventaja cobra especial valor cuando el conjunto de datos disponible es reducido, algo frecuente en aplicaciones de visión artificial muy específicas. En esos escenarios, entrenar una red profunda desde cero es arriesgado: al tener muchos parámetros y pocos ejemplos, la red tiende al sobreajuste (overfitting), es decir, memoriza el conjunto de entrenamiento en lugar de aprender patrones que generalicen a datos nuevos [8].

El fundamento de la técnica proviene de una observación sobre cómo aprenden las redes convolucionales. Sus primeras capas tienden a capturar rasgos visuales de bajo nivel —bordes, texturas, gradientes— útiles para casi cualquier problema de visión, sin importar el dominio. Las capas más profundas, en cambio, aprenden representaciones de mayor nivel de abstracción, ligadas al dominio concreto en que se entrenaron [25]. Gracias a ello es posible conservar (congelar) las capas iniciales de un modelo preentrenado y reemplazar únicamente las finales por una cabeza clasificadora diseñada para el nuevo problema.

En la práctica, la adaptación de un modelo preentrenado suele realizarse en dos fases [24]:

1. **Extracción de características (Feature Extraction).** Se congela por completo la red base preentrenada y se entrena solo la cabeza clasificadora añadida. Así, el modelo aprende a asociar las representaciones que ya posee con las nuevas clases objetivo, y lo hace en pocas épocas.

2. **Ajuste fino (Fine-tuning).** Una vez que la cabeza se ha estabilizado, se descongelan las últimas capas de la red base y se continúa el entrenamiento con una tasa de aprendizaje muy baja. De este modo, esas capas se adaptan de forma gradual a las particularidades del nuevo dominio sin borrar el conocimiento adquirido previamente [8].

2.5 MobileNetV2

MobileNetV2 es una arquitectura de red neuronal convolucional desarrollada por investigadores de Google y presentada en la conferencia CVPR 2018 como sucesora de MobileNetV1 [29], [26]. Su diseño responde a una necesidad muy concreta dentro de la visión por computadora: llevar el aprendizaje profundo a dispositivos con recursos limitados —teléfonos móviles, sistemas embebidos o cámaras inteligentes—, donde no se cuenta con la memoria ni la capacidad de cómputo de una estación con GPU dedicada. Para lograrlo, la arquitectura reduce drásticamente el número de operaciones y de parámetros respecto a las redes convolucionales tradicionales, sin sacrificar de forma significativa la precisión.

El elemento que hace posible esa eficiencia es la convolución separable en profundidad (depthwise separable convolution), que descompone una convolución estándar en dos operaciones más económicas: una que filtra cada canal por separado y otra que combina la

información entre canales. Sobre esa base, MobileNetV2 introduce dos innovaciones arquitectónicas principales:

- **Bloques residuales invertidos (Inverted Residuals).** A diferencia de los bloques residuales clásicos, que comprimen la dimensionalidad en las capas intermedias y la expanden en los extremos, MobileNetV2 hace lo inverso: expande la representación en el interior del bloque antes de aplicar la convolución separable en profundidad y la vuelve a comprimir al final. Esta inversión permite procesar la información en un espacio de mayor dimensión —donde se conservan más detalles— manteniendo ligeras las conexiones entre bloques [26].

- **Cuellos de botella lineales (Linear Bottlenecks).** En la capa de proyección final de cada bloque se elimina la activación no lineal (ReLU). El motivo es que, cuando una representación tiene pocas dimensiones, aplicar ReLU descarta información que después no puede recuperarse; al dejar esa capa lineal se evita esa pérdida y se preserva la capacidad expresiva del bloque [26].

Como la mayoría de las arquitecturas modernas, MobileNetV2 se utiliza preentrenada. Su versión pública aprendió sobre el subconjunto ImageNet ILSVRC-2012 [27], compuesto por cerca de 1.28 millones de imágenes repartidas en 1,000 categorías —el ImageNet completo es todavía mayor, con más de 14 millones de imágenes en unas 20,000 categorías—. Entrenar sobre semejante volumen de datos permite que la red aprenda a reconocer una enorme variedad de formas, texturas y patrones visuales de propósito general; ese "saber ver" acumulado es justamente lo que se aprovecha al emplearla como base para Transfer Learning en problemas con datasets pequeños, como el de este trabajo.

En este proyecto, MobileNetV2 se utilizó como red base del modelo CNN mejorado, siguiendo la estrategia de entrenamiento en dos fases. Durante la primera, la red base se congeló por completo para que actuara únicamente como extractor de características, mientras se entrenaba una cabeza clasificadora propia —formada por capas GlobalAveragePooling2D, Dense y Dropout— encargada de distinguir las seis emociones de interés. En la segunda, se descongelaron sus últimas 30 capas y se realizó el ajuste fino con una tasa de aprendizaje muy baja (2×10^{-5}), de modo que esas capas se adaptaran a las particularidades del reconocimiento de expresiones faciales bajo las condiciones controladas del estudio, sin destruir lo aprendido sobre ImageNet.

2.6 Visión Artificial

La visión artificial, también llamada visión por computadora (computer vision), es la rama de la inteligencia artificial que se ocupa de dotar a las máquinas de la capacidad de interpretar, analizar y comprender la información visual del mundo real a partir de imágenes o video. Su

propósito es que un sistema computacional realice tareas que tradicionalmente exigían percepción humana, como reconocer objetos, seguir movimientos, identificar rostros o interpretar expresiones faciales [2].

Para lograrlo, se apoya en una combinación de procesamiento digital de imágenes, estadística, geometría computacional y aprendizaje automático. El flujo de trabajo típico de un sistema de este tipo comprende varias etapas: la adquisición de la imagen, el preprocesamiento (por ejemplo, reducción de ruido y ajuste de contraste), la segmentación o identificación de las regiones relevantes, la extracción de características y, finalmente, la clasificación o el reconocimiento [2].

Los avances recientes en aprendizaje profundo, y en particular el uso de redes neuronales convolucionales (CNN), han elevado la precisión de la visión artificial hasta niveles comparables —o incluso superiores— a los humanos en tareas específicas, ya que estas redes extraen automáticamente patrones visuales complejos sin necesidad de diseñar los atributos a mano. Ese salto ha impulsado su aplicación en numerosos campos:

- **Vehículos autónomos**, donde modelos de visión reconocen señales de tránsito, peatones y obstáculos en tiempo real.
- **Medicina**, con sistemas que analizan imágenes radiológicas para detectar enfermedades como el cáncer o la retinopatía.
- **Seguridad**, mediante videovigilancia capaz de identificar comportamientos sospechosos o personas por reconocimiento facial.
- **Robótica social**, como en este proyecto, donde se emplean imágenes faciales para inferir el estado emocional del usuario y ajustar la respuesta del sistema [20].

En este trabajo, la visión artificial es la tecnología que habilita todo el reconocimiento emocional. A partir de la imagen capturada por la cámara se extraen los puntos clave del rostro con MediaPipe y, después, se clasifican mediante los modelos entrenados en TensorFlow. Esa integración es la que permite que el sistema interprete la expresión de una persona en tiempo real y adapte su comportamiento en consecuencia.

2.7 Robótica Social

La robótica social es la subdisciplina de la robótica dedicada a diseñar y estudiar robots capaces de interactuar con las personas de forma natural y significativa. Se distingue con claridad de la robótica industrial: mientras esta última opera en entornos estructurados ejecutando tareas técnicas o repetitivas, los robots sociales están pensados para convivir en espacios humanos, donde deben percibir señales sociales, responder a ellas y ajustarse a las normas culturales y al contexto [3].

Para emular esas habilidades sociales, estos sistemas integran aportes de varias disciplinas —inteligencia artificial, percepción computacional, psicología, diseño de interfaces, lingüística y ciencias cognitivas—. Entre las capacidades que suelen caracterizarlos se encuentran:

- El reconocimiento de emociones y expresiones faciales.
- La comprensión del lenguaje natural y la generación de respuestas.
- El mantenimiento del contacto visual, la gesticulación y la expresión de emociones.
- La adaptación de su comportamiento según el usuario y la situación.

Su campo de aplicación es amplio y sigue creciendo:

- Educación personalizada, con tutores inteligentes que reaccionan a las emociones del estudiante.
- Atención a adultos mayores, ofreciendo compañía y un monitoreo poco intrusivo.
- Intervenciones terapéuticas, sobre todo en contextos de autismo o rehabilitación emocional.
- Entornos públicos, como asistentes en museos, aeropuertos o centros comerciales [22].

Para que un robot social resulte aceptado y eficaz no basta con que funcione: debe ser creíble, legible, empático y adaptable. En otras palabras, sus comportamientos deben ser comprensibles para las personas, sus respuestas emocionales apropiadas, y su conducta capaz de amoldarse a cada usuario o grupo con el que interactúa [22].

Este trabajo aborda la robótica social desde su componente perceptual: un sistema que detecta expresiones faciales en tiempo real y sienta las bases para que un agente automatizado responda según el estado emocional del usuario. La combinación de visión artificial, aprendizaje profundo y modelos de reconocimiento de emociones es lo que permite esa sensibilidad afectiva, condición necesaria para una interacción más cercana y humana.

2.8 Interacción Humano-Robot

La interacción humano-robot (HRI, por sus siglas en inglés) es el campo interdisciplinario que estudia y diseña la comunicación entre las personas y los sistemas robóticos. Su meta es que esa comunicación sea a la vez efectiva, natural y segura, para lo cual debe atender aspectos técnicos, cognitivos, sociales y emocionales. La HRI abarca desde las interfaces físicas y visuales hasta el reconocimiento de gestos, voz, emociones y comportamientos, de modo que el robot pueda responder de forma comprensible y adaptada a cada situación [4].

A medida que los robots sociales se incorporan a contextos cotidianos —la educación, la salud o la asistencia personal—, la calidad de esta interacción se vuelve determinante para su aceptación y utilidad. En este proyecto, la detección de emociones mediante visión artificial es el mecanismo perceptual con el que se busca enriquecer esa interacción, al darle al sistema información sobre el estado afectivo de la persona con la que se relaciona.

2.9 Reconocimiento de Emociones

El reconocimiento de emociones es una disciplina enmarcada en la inteligencia artificial afectiva (affective computing) cuyo objetivo es identificar, clasificar e interpretar los estados emocionales de una persona a partir de señales observables. Esas señales pueden ser de distinta naturaleza —expresiones faciales, tono de voz, lenguaje corporal o datos fisiológicos como el ritmo cardíaco—, y de la modalidad elegida depende buena parte del enfoque técnico. Cuando el análisis se apoya en la visión artificial, como en este caso, se centra en los rasgos del rostro captados en imágenes o video, y emplea algoritmos de aprendizaje automático o profundo para reconocer los patrones asociados a las emociones de interés: felicidad, tristeza, enojo, asombro, disgusto y neutralidad [5].

Estos sistemas tienen hoy una presencia amplia en campos como la robótica social, la educación personalizada, la salud mental y la interacción humano-máquina. En este proyecto, el reconocimiento se realiza a partir de imágenes faciales y se aborda con dos enfoques complementarios: modelos que procesan la imagen directamente (redes convolucionales) y modelos que operan sobre características geométricas del rostro extraídas con MediaPipe, todos implementados con TensorFlow y scikit-learn, según se detalla en el Capítulo 3.

2.10 Expresiones Faciales

Las expresiones faciales son manifestaciones visibles de los estados emocionales internos, producidas por movimientos específicos de los músculos del rostro. Constituyen una de las formas de comunicación no verbal más espontáneas y universales entre las personas, capaces de transmitir información afectiva de manera inmediata. Las investigaciones pioneras de Paul Ekman identificaron seis emociones básicas universales —felicidad, tristeza, enojo, miedo, sorpresa y asco (disgusto)—, cada una asociada a patrones musculares faciales reconocibles a través de distintas culturas [6]. En este trabajo se partió de ese marco, pero se adaptó a seis clases de interés para la aplicación —felicidad, tristeza, enojo, asombro, disgusto y neutralidad—, sustituyendo el miedo por un estado neutral, dada su mayor pertinencia en escenarios de interacción humano-robot.

Dentro de la visión artificial, estas expresiones funcionan como indicadores para inferir el estado afectivo de una persona a partir de imágenes o secuencias de video. Su análisis

automático hace posible que un sistema interprete la dimensión emocional de una interacción, algo especialmente valioso en aplicaciones de robótica social, educación, atención médica y seguridad [6].

2.11 Fundamentos de Expresiones Faciales (FACS)

El Facial Action Coding System (FACS) es un sistema de codificación creado por Paul Ekman y Wallace V. Friesen que describe de forma objetiva los movimientos del rostro mediante combinaciones de unidades de acción (AUs, por sus siglas en inglés), cada una correspondiente a la contracción de un músculo facial específico. Su utilidad radica en que permite descomponer cualquier expresión, por compleja que sea, en un conjunto de elementos básicos y observables, razón por la cual se ha empleado tanto en la investigación psicológica como en aplicaciones tecnológicas de reconocimiento emocional.

Bajo este esquema, cada emoción básica del modelo de Ekman —felicidad, tristeza, enojo, miedo, sorpresa y disgusto— se asocia con un patrón característico de AUs; la neutralidad, en cambio, no es una emoción básica, sino la ausencia de una activación muscular marcada, y por eso se incluye como clase de referencia en este trabajo. Un ejemplo típico es el asombro, que suele codificarse con la elevación de las cejas (AU1 + AU2) y la apertura de la boca (AU26 o AU27). Esta base anatómica ha facilitado el entrenamiento de sistemas de visión artificial, al ofrecer un vínculo claro entre los movimientos musculares y las emociones que representan.

En este proyecto, el FACS funciona como marco de referencia para interpretar los puntos faciales (landmarks) que entrega MediaPipe, y ayuda a relacionar las variaciones geométricas del rostro con el estado emocional correspondiente [6].

2.12 Extracción de Características Faciales

La extracción de características faciales es una de las etapas centrales de cualquier sistema de visión artificial orientado al reconocimiento facial o emocional. Consiste en identificar y cuantificar los puntos o regiones del rostro que concentran la información útil sobre su geometría, forma, textura o movimiento muscular, de modo que una imagen —que en bruto es solo una matriz de píxeles— quede representada por un conjunto de valores manejables y descriptivos. Estas características pueden ser de dos tipos: geométricas, cuando se basan en medidas espaciales como las distancias entre los ojos, la posición de las cejas o los ángulos de la mandíbula; o de apariencia, cuando se derivan de la intensidad de los píxeles, los gradientes y las texturas de la imagen.

En las aplicaciones actuales, esta etapa suele automatizarse con herramientas especializadas. En este proyecto se emplea MediaPipe Face Mesh, que detecta de forma automática 468 puntos faciales (landmarks) y proporciona así una representación estructurada y detallada del rostro. Las características obtenidas sirven después como entrada para los modelos de clasificación: los valores geométricos alimentan a los modelos basados en características (SVM, Random Forest y el MLP), mientras que la imagen completa se procesa con las redes neuronales convolucionales, todo con el fin de asignar a cada rostro la emoción que expresa [7].

2.13 Detección

En visión artificial y aprendizaje automático, la detección es el proceso por el cual un sistema determina la presencia o ausencia de un objeto, evento o característica dentro de una imagen, un video o un conjunto de datos, y establece su ubicación. Conviene distinguirla de la identificación: mientras esta última se pregunta qué o quién es el objeto, la detección responde a una pregunta previa y más básica —si el objeto está y dónde se encuentra—.

En el análisis facial, detectar puede significar localizar los rostros presentes en una imagen, ubicar los puntos clave del rostro (landmarks) o delimitar regiones concretas como los ojos, la boca o las cejas. Se trata de un paso decisivo, porque acota el área de interés que luego será analizada o clasificada por un modelo más complejo; sin una detección fiable, las etapas posteriores trabajarían sobre información irrelevante o incompleta [2].

2.14 Identificación

La identificación es el proceso mediante el cual un sistema reconoce una entidad observada y la asocia con una clase, etiqueta o categoría ya conocida. En visión artificial suele implicar el análisis de características visuales concretas —de un rostro, un objeto o un patrón— para determinar qué representa la imagen o a quién corresponde. A diferencia del reconocimiento de patrones en sentido amplio, que puede quedarse en una simple detección o clasificación, la identificación establece una correspondencia específica con una identidad o un estado particular. En este trabajo, ese estado es la emoción del rostro: el sistema no solo detecta que hay una cara, sino que la asocia con una de las seis categorías emocionales definidas [18].

2.15 Algoritmo de Visión Artificial

Un algoritmo de visión artificial es el conjunto de instrucciones computacionales que permite a un sistema interpretar, procesar y extraer información significativa de imágenes o secuencias de video. Bajo ese término caben operaciones muy diversas —detección de

bordes, segmentación, extracción de características, reconocimiento de patrones o clasificación de objetos y expresiones—, que suelen encadenarse para transformar los píxeles de entrada en una decisión o una etiqueta. Cuando estos algoritmos se combinan con el aprendizaje automático, incorporan modelos entrenables, como las redes neuronales convolucionales, capaces de aprender a reconocer estructuras visuales complejas a partir de ejemplos, en lugar de depender de reglas fijas.

En este proyecto, la parte de visión artificial se apoya en MediaPipe para la detección de los puntos faciales (landmarks) y en los modelos de clasificación desarrollados con TensorFlow y scikit-learn, encargados de asignar a cada rostro su emoción. La combinación de ambos permite evaluar el estado afectivo de una persona directamente a partir de su expresión facial [2].

2.16 Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son un tipo de red de aprendizaje profundo diseñado específicamente para trabajar con datos que tienen estructura espacial, como las imágenes o el video. Su arquitectura se inspira en la organización de la corteza visual de los mamíferos, lo que le permite extraer características de forma jerárquica: de lo simple a lo complejo, capa tras capa [8].

Una CNN se compone de una secuencia de capas especializadas, cada una con una función definida:

- Capas convolucionales: aplican filtros (kernels) entrenables que recorren la imagen para detectar rasgos espaciales como bordes, esquinas o texturas locales.
- Funciones de activación (como ReLU): introducen no linealidad, sin la cual la red solo podría aprender relaciones lineales y no representaciones complejas.
- Capas de agrupamiento (pooling): reducen la dimensión espacial conservando la información más relevante y aportan cierta invariancia frente a pequeños desplazamientos.
- Capas totalmente conectadas: interpretan las características extraídas y producen la clasificación final.
- Técnicas de regularización (como Dropout o Batch Normalization): mitigan el sobreajuste y estabilizan el entrenamiento [8], [9].

El entrenamiento de una CNN consiste en minimizar una función de pérdida, normalmente mediante la retropropagación del error y el descenso del gradiente, con optimizadores como SGD o Adam que ajustan los parámetros de manera eficiente [8]. Frente a las redes densas

tradicionales, una de sus grandes ventajas es la compartición de pesos: un mismo filtro se reutiliza a lo largo de toda la imagen, lo que reduce drásticamente el número de parámetros y acelera el aprendizaje. A esto se suma su carácter jerárquico, que le permite reconocer desde patrones simples, como líneas, hasta estructuras completas, como un rostro [9].

En visión por computadora, las CNN se han consolidado como la arquitectura dominante en tareas como:

- Clasificación de imágenes.
- Detección de objetos.
- Segmentación semántica.
- Reconocimiento facial.
- Reconocimiento de emociones faciales, donde permiten captar cambios sutiles en la expresión del rostro [10].

Esa misma capacidad es la que las vuelve idóneas para este trabajo: analizan directamente los patrones visuales del rostro asociados a cada emoción, lo que las convierte en uno de los cinco modelos evaluados en el sistema de reconocimiento emocional.

2.17 Comparación entre Redes Neuronales Convolucionales (CNN) y otros modelos de clasificación

Las redes neuronales convolucionales (CNN) se han consolidado como una de las arquitecturas más efectivas para el procesamiento de imágenes, gracias a su capacidad de extraer de forma jerárquica las características espaciales relevantes sin necesidad de diseñarlas a mano. Sin embargo, no son la única opción para una tarea de clasificación, y resulta pertinente contrastar su desempeño con el de enfoques más tradicionales del aprendizaje automático, como las Máquinas de Vectores de Soporte (SVM) y los Bosques Aleatorios (Random Forest), que este trabajo también evalúa.

Las SVM buscan la frontera de decisión que separa las clases con el mayor margen posible; gracias al uso de funciones kernel, pueden trazar fronteras no lineales proyectando los datos a espacios de mayor dimensión. Esa propiedad las ha hecho muy útiles en clasificación facial con pocas muestras, donde suelen sobreajustar menos que una red profunda. El Random Forest, en cambio, es un método de conjunto (ensemble): combina las predicciones de numerosos árboles de decisión entrenados sobre subconjuntos aleatorios de los datos y de las características, de modo que la agregación reduce la varianza del modelo, lo vuelve robusto frente al ruido y, además, ofrece cierta interpretabilidad a través de la importancia de cada característica.

La diferencia de fondo entre estos métodos y las CNN está en el manejo de las características. Tanto la SVM como el Random Forest requieren que un diseñador defina y extraiga de antemano las características que describen cada muestra —una etapa manual que en dominios complejos puede ser costosa y difícil de escalar—. Las CNN, por el contrario, integran ese paso en el propio entrenamiento y aprenden las características directamente de los píxeles, lo que les permite capturar patrones visuales más abstractos.

La literatura reporta que, cuando se dispone de grandes volúmenes de datos y de recursos de cómputo adecuados, las CNN superan de manera consistente a estos modelos clásicos en tareas como la clasificación emocional a partir de imágenes faciales [16]. No obstante, esa ventaja no es absoluta: en conjuntos de datos pequeños y controlados —como el de este trabajo—, los modelos basados en características geométricas pueden alcanzar un desempeño competitivo, hipótesis que se examina en el capítulo de resultados.

2.18 Clasificación Multiclase

Cuando un problema de aprendizaje supervisado exige elegir una etiqueta entre tres o más categorías mutuamente excluyentes, hablamos de clasificación multiclase. Su diferencia con la variante binaria radica en la cantidad de resultados posibles: mientras esta última solo separa dos alternativas opuestas (por ejemplo, "positivo" frente a "negativo"), la multiclase obliga al modelo a decidir cuál de varias etiquetas describe mejor cada instancia, sin que dos de ellas puedan ser correctas a la vez. El sistema desarrollado en este trabajo se enmarca precisamente en ese esquema, ya que debe asignar cada rostro capturado a una de las seis expresiones consideradas: felicidad, tristeza, enojo, asombro, disgusto o neutralidad. Cada una de esas expresiones constituye una clase independiente, de modo que la salida del clasificador nunca reparte una imagen entre dos emociones simultáneas.

La forma de resolver esta tarea varía según la familia de algoritmos empleada. En las redes neuronales convolucionales, el aprendizaje ocurre a partir de un conjunto de imágenes etiquetadas: las capas convolucionales extraen de manera automática los patrones visuales que caracterizan a cada expresión y ajustan sus parámetros hasta discriminar entre las seis categorías. Para traducir la actividad de la última capa en una decisión, se recurre a la función de activación softmax, que convierte los valores de salida en una distribución de probabilidad sobre las clases y permite interpretar cuál de ellas resulta más verosímil para la entrada analizada. Durante el entrenamiento, el error se cuantifica mediante la entropía cruzada categórica, cuya penalización crece a medida que la probabilidad asignada a la etiqueta verdadera se aleja de la unidad, guiando así el ajuste de los pesos de la red [9].

Conviene señalar que esta lógica de una etiqueta por instancia también gobierna a los demás modelos evaluados en la investigación. Los cinco enfoques comparados (la CNN base, la versión con Transfer Learning basada en MobileNetV2, el perceptrón multicapa alimentado

con las 1404 coordenadas de los landmarks faciales, la máquina de soporte vectorial y el bosque aleatorio) abordan el mismo desafío de asignar una única emoción a cada muestra. La diferencia está en el mecanismo interno con que cada uno construye esa frontera de decisión, pero todos comparten la naturaleza multiclase del problema y la exigencia de separar correctamente las seis expresiones faciales que el sistema busca reconocer.

2.19 Preprocesamiento de Imágenes

Antes de que cualquier modelo reciba una imagen, esta debe pasar por una etapa de acondicionamiento que corrige las irregularidades propias de la captura y deja los datos en una forma predecible para el análisis. El preprocesamiento agrupa justamente ese conjunto de operaciones que se aplican sobre la imagen cruda para elevar su calidad y uniformarla, de modo que las diferencias que llegan al clasificador respondan al contenido y no a las condiciones en que se tomó la fotografía. Cuando las entradas comparten la misma escala, el mismo rango de intensidades y un formato consistente, el algoritmo de aprendizaje puede concentrarse en los patrones que realmente distinguen una expresión de otra en lugar de compensar variaciones espurias [15].

Entre las transformaciones más habituales se encuentran las siguientes, cada una atendiendo un problema distinto de la señal de entrada:

- Redimensionamiento: se ajusta la imagen a una resolución fija para que todas ocupen las mismas dimensiones. En este trabajo el rostro se lleva a 96×96 píxeles, tamaño que reciben tanto la CNN base como la CNN mejorada con MobileNetV2.
- Conversión a escala de grises: se descarta la información de color cuando no aporta al problema, reduciendo la imagen a un solo canal de luminancia. La CNN base opera precisamente sobre imágenes de 96×96 en gris, mientras que la variante con MobileNetV2 conserva los tres canales RGB.
- Normalización de los valores de píxel: se reescalan las intensidades a un rango acotado para que la magnitud de los números no domine el ajuste de los pesos y el entrenamiento converja de manera más estable.
- Reducción de ruido: se filtran las perturbaciones introducidas por el sensor o la iluminación, evitando que el modelo aprenda artefactos en vez de rasgos faciales.
- Aumento de datos (data augmentation): se generan variantes sintéticas a partir de las imágenes existentes mediante giros, desplazamientos o cambios de brillo, algo especialmente valioso aquí porque el conjunto proviene de un único sujeto fotografiado en condiciones controladas y conviene ampliar su diversidad aparente.

El propósito de fondo es estandarizar la entrada y aligerar la tarea de aprendizaje: al recortar la complejidad que enfrenta el modelo y homogeneizar lo que percibe, se mejora la eficiencia del proceso y se favorece que las seis expresiones consideradas se separen con mayor claridad. Conviene aclarar que este acondicionamiento aplica a los modelos que trabajan directamente sobre píxeles; los enfoques basados en las 468 marcas faciales de MediaPipe siguen una preparación distinta, orientada a coordenadas y no a la imagen completa.

2.20 Base de Datos

Toda base de datos responde a la necesidad de guardar información de forma ordenada, de modo que pueda consultarse, modificarse y actualizarse sin ambigüedades ni pérdidas. Su diseño busca que los datos permanezcan íntegros y consistentes a lo largo del tiempo, y que sigan disponibles cuando un programa o un usuario los solicite. Para lograrlo, la información no se almacena de cualquier manera, sino conforme a un modelo que define cómo se relacionan los elementos entre sí; de ahí que existan bases relacionales, apoyadas en tablas vinculadas por llaves, así como esquemas no relacionales, jerárquicos o documentales, cada uno pensado para un tipo distinto de carga de trabajo y de patrón de consulta.

Cuando el problema pasa por la inteligencia artificial y la visión por computadora, la base de datos deja de ser un simple depósito y se vuelve la materia prima del aprendizaje: sin colecciones de ejemplos correctamente etiquetados, un modelo carece de referencia para ajustar sus parámetros y reconocer regularidades. Esa dependencia explica por qué la calidad y la organización de los datos condicionan directamente el desempeño final de cualquier clasificador [17]. En este trabajo, dicha materia prima consistió en un acervo aproximado de 900 imágenes capturadas de un solo sujeto bajo condiciones controladas de estudio fotográfico, agrupadas en las seis clases de estudio: felicidad, tristeza, enojo, asombro, disgusto y neutralidad. Ese repositorio alimentó dos rutas de procesamiento distintas. Por un lado, las redes convolucionales consumieron directamente las imágenes; por otro, a partir de cada rostro se derivaron representaciones numéricas mediante MediaPipe Face Mesh, que entrega 468 puntos de referencia en tres dimensiones. De esa malla se obtuvieron los 1404 valores que recibe el perceptrón multicapa, correspondientes a las coordenadas x, y, z de cada landmark, así como las 11 características geométricas que emplean los modelos SVM y Random Forest. Organizar y etiquetar con cuidado esos conjuntos fue lo que permitió entrenar y validar los distintos algoritmos de manera comparable.

2.21 TensorFlow

Google liberó TensorFlow como una biblioteca de código abierto pensada para construir y ejecutar de manera eficiente modelos de aprendizaje automático y de aprendizaje profundo. Su diseño gira en torno a los tensores, arreglos multidimensionales con los que se representan

grandes cantidades de datos numéricos; esa forma de organizar la información resulta muy conveniente para problemas de visión por computadora, procesamiento de lenguaje natural y reconocimiento de voz [9]. Al trabajar con estas estructuras, la biblioteca describe cada cálculo como una operación sobre tensores, lo que facilita paralelizar el entrenamiento y aprovechar hardware especializado.

Entre las cualidades que explican su adopción destaca la capacidad de correr el mismo modelo en distintas plataformas de cómputo. TensorFlow puede ejecutarse sobre CPU para desarrollo y pruebas, sobre GPU cuando se requiere acelerar el entrenamiento de redes profundas, e incluso sobre TPU (Tensor Processing Units), circuitos diseñados por Google específicamente para operaciones tensoriales. Esta portabilidad, unida a su escalabilidad, permite pasar de un prototipo en una computadora personal a un despliegue sobre servidores sin reescribir el modelo, algo valioso cuando la carga computacional crece.

A partir de la versión 2.x, la biblioteca incorpora de forma nativa a Keras como interfaz de alto nivel, con lo que definir, entrenar y poner en funcionamiento una red neuronal se vuelve una tarea de sintaxis clara y modular. Gracias a esta integración fue posible armar los modelos del sistema como arquitecturas secuenciales o funcionales, vigilar el avance del entrenamiento mediante callbacks y registrar métricas para inspeccionarlas con herramientas como TensorBoard, además de exportar los modelos ya entrenados para reutilizarlos fuera del entorno de desarrollo, sea en producción o en equipos móviles.

Otro aspecto que conviene señalar es la flexibilidad del modo de programación. TensorFlow admite un estilo imperativo mediante eager execution, donde cada operación se evalúa de inmediato y el código se depura con facilidad, y también la construcción de gráficas computacionales estáticas, que optimizan la ejecución en entornos de alto rendimiento. A esto se suman utilidades para ajustar automáticamente los parámetros del modelo, para gestionar el flujo de datos de entrada con tf.data y para llevar los modelos a otros destinos mediante TensorFlow Lite o TensorFlow.js [9].

Para este trabajo, TensorFlow junto con Keras sirvió de base para desarrollar y entrenar los modelos de aprendizaje profundo encargados de clasificar expresiones faciales. Sobre este framework se construyó la red neuronal convolucional base, que procesa imágenes de 96×96 píxeles en escala de grises con tres capas Conv2D, así como la versión mejorada apoyada en Transfer Learning con MobileNetV2, que recibe imágenes de 96×96 en RGB. También se implementó con TensorFlow el perceptrón multicapa (MLP), cuya entrada son los 1404 valores que resultan de los 468 puntos de referencia faciales entregados por MediaPipe, cada uno con sus coordenadas x, y, z. La salida de estos modelos corresponde a una de las seis categorías consideradas, entre ellas felicidad, tristeza y enojo. Apoyarse en TensorFlow permitió obtener una implementación ordenada y capaz de escalar dentro del sistema de reconocimiento de emociones.

2.22 Keras

Dentro del ecosistema de TensorFlow, Keras funciona como la capa de programación de alto nivel con la que se describen y entrenan redes neuronales sin tener que manipular directamente las operaciones de bajo nivel del motor de cálculo. Fue creada inicialmente por François Chollet [28] y, a partir de la versión 2.x de TensorFlow, quedó incorporada como su interfaz oficial, de modo que hoy se emplea como la puerta de entrada habitual para el modelado con esta plataforma [12]. Su valor práctico radica en que reduce la distancia entre la idea de una arquitectura y su implementación, ya que expone bloques reutilizables (capas, funciones de activación, optimizadores y funciones de pérdida) que se ensamblan con muy pocas líneas de código.

La biblioteca ofrece dos formas complementarias de definir un modelo. La interfaz secuencial permite apilar capas una tras otra cuando el flujo de datos es lineal, mientras que la interfaz funcional habilita topologías más elaboradas, con ramificaciones o entradas múltiples. Esta flexibilidad resulta conveniente para pasar de prototipos rápidos a diseños más exigentes sin cambiar de herramienta, y se complementa con utilidades para regularizar el entrenamiento, evaluar el desempeño y guardar los pesos aprendidos.

Para el sistema de reconocimiento de expresiones faciales desarrollado en esta tesis, Keras es la herramienta con la que se construyen y entrenan los modelos basados en aprendizaje profundo. Con ella se implementa la red convolucional base, que recibe imágenes de 96×96 en escala de grises y las procesa mediante tres capas Conv2D; también se arma la versión mejorada apoyada en Transfer Learning con MobileNetV2, que opera sobre imágenes de 96×96 en RGB. Del mismo modo, el perceptrón multicapa se define en Keras para clasificar el vector de 1404 valores que proviene de los 468 puntos de referencia faciales entregados por MediaPipe, cada uno con sus coordenadas x, y, z. Los modelos restantes, la máquina de soporte vectorial y el bosque aleatorio, se apoyan en scikit-learn y quedan fuera de esta biblioteca. De esta manera, Keras concentra la parte del proyecto que requiere entrenar redes neuronales sobre las seis clases de salida contempladas, aportando una vía ordenada para experimentar con las distintas configuraciones antes de llevarlas a la aplicación en tiempo real.

2.23 MediaPipe

MediaPipe corresponde a una biblioteca de código abierto liberada por Google cuyo propósito es armar tuberías de percepción multimodal que operen sobre datos en vivo. Su diseño responde a las necesidades de tareas de visión por computadora, aplicaciones de realidad aumentada y desarrollos de robótica, donde el flujo de imágenes debe procesarse sin demoras perceptibles. El planteamiento es marcadamente modular: la herramienta agrupa capacidades de detección, segmentación, seguimiento y estimación bajo un mismo entorno

afinado para el rendimiento, de modo que la ejecución sigue siendo viable aun cuando el hardware disponible es modesto o cuenta con memoria y cómputo acotados [13]. Esa eficiencia resulta determinante para un sistema que, como el desarrollado en este trabajo, aspira a reconocer expresiones faciales mientras la persona interactúa con el robot.

El funcionamiento interno se apoya en un grafo de procesamiento de datos, un esquema conocido como graph-based computation. Cada nodo de ese grafo equivale a un calculador, es decir, un módulo encargado de aplicar una operación concreta sobre la información que recibe, y el resultado se propaga hacia los siguientes nodos siguiendo las conexiones definidas. Organizar el trabajo de esta manera trae ventajas prácticas: los módulos pueden reaprovecharse entre proyectos, distintas etapas llegan a ejecutarse en paralelo y varios modelos de aprendizaje profundo se acoplan sin fricción dentro del mismo flujo. El efecto conjunto es una reducción de la latencia que habilita el análisis cuadro a cuadro en tiempo real. A ello se suma una amplia cobertura de plataformas, pues la biblioteca funciona en Android, en iOS, en entornos de escritorio bajo Windows, Linux y macOS, e incluso dentro del navegador web gracias a WebAssembly o a TensorFlow.js [13]. Para el reconocimiento facial planteado aquí, la utilidad más directa proviene de su componente Face Mesh, que localiza 468 puntos de referencia tridimensionales del rostro y entrega así las coordenadas que alimentan a los clasificadores basados en geometría del sistema.

2.24 MediaPipe Face Mesh

Dentro de la biblioteca, el componente que concentra mayor sofisticación es Face Mesh, capaz de localizar y seguir 468 puntos faciales en tres dimensiones a partir de una simple imagen RGB, sin cámaras de profundidad ni marcadores adheridos a la piel. Esa densidad de puntos ofrece un mapa geométrico mucho más fino que el de propuestas anteriores como Dlib, que trabaja con 68 puntos, u OpenFace, cuyo rango va de 68 a 194; el resultado es una descripción notablemente más rica de la superficie del rostro [14].

Su funcionamiento se apoya en dos redes que operan encadenadas: una primera etapa detecta el rostro con rapidez y una segunda, de regresión, refina la ubicación de cada landmark hasta alcanzar buena exactitud. Ambas fueron pensadas para ser ligeras y ejecutarse en hardware móvil, de modo que el modelo prescinde de equipo especializado y se vuelve viable en plataformas portátiles o embebidas [14].

Aplicaciones del modelo

El abanico de usos de Face Mesh es amplio y abarca, entre otros:

- Reconocimiento de expresiones faciales y de estados emocionales.
- Seguimiento de la mirada y medición de la atención visual.

- Animación de rostros y generación de avatares en realidad aumentada.
- Interacción humano-computadora (HCI) y robótica social.
- Evaluación biométrica con fines médicos y psicológicos.

Gracias a un margen de error de apenas uno o dos píxeles en promedio y a su bajo consumo de recursos, esta herramienta se ha llevado a productos comerciales, aplicaciones móviles, videojuegos y entornos de educación interactiva [13].

Integración con sistemas de aprendizaje profundo

MediaPipe no emite por sí mismo una etiqueta de clase; su papel es el de una etapa previa que extrae rasgos visuales aprovechables por modelos posteriores. Las coordenadas que entrega pueden convertirse en vectores geométricos o en mapas de calor y luego alimentar redes convolucionales (CNN) o arquitecturas recurrentes como RNN o LSTM, según la tarea sea clasificar emociones, anticipar conductas o verificar identidades [12].

Para este trabajo, Face Mesh cumplió la función de extraer los puntos del rostro sobre las imágenes capturadas en tiempo real. Con los 468 landmarks y sus tres coordenadas (x, y, z) se construyó un vector de 1404 valores que se entregó directamente al perceptrón multicapa (MLP), mientras que esos mismos puntos sirvieron para calcular las 11 características geométricas con las que operan el SVM y el Random Forest. La malla facial fue, por tanto, el insumo común de los modelos basados en geometría, en tanto que las redes convolucionales trabajaron sobre la imagen de 96×96 y no sobre los landmarks. Esa separación permitió comparar, dentro de un mismo sistema de reconocimiento de seis clases (felicidad, tristeza, enojo, asombro, disgusto y neutralidad), enfoques que parten de representaciones muy distintas de un mismo rostro.

Ventajas adicionales

- Eficiencia de cómputo: al estar implementada en C++, la biblioteca corre con baja latencia y alcanza hasta 30 cuadros por segundo en teléfonos móviles.
- Escalabilidad: admite combinar varios modelos —rostro, manos y cuerpo— dentro de un mismo flujo de procesamiento sin degradar el rendimiento.
- Compatibilidad modular: se acopla con otras bibliotecas como OpenCV, TensorFlow o PyTorch para ampliar sus alcances.
- Respaldo de una comunidad activa y documentación oficial extensa, lo que facilita su adopción tanto en proyectos académicos como en entornos de producción [13], [14].

2.25 Evaluación

Evaluar significa someter un objeto de estudio (un sistema, un método, un modelo o una intervención) a un examen ordenado que permita pronunciarse sobre su calidad y su desempeño real. No se trata de una impresión subjetiva, sino de un juicio que se sostiene en evidencia: se recogen datos, se procesan y se contrastan contra criterios o estándares fijados de antemano, de modo que la conclusión pueda justificarse y, sobre todo, reproducirse. Según el enfoque adoptado, la valoración puede ser cuantitativa, cualitativa o combinar ambos registros, y su alcance abarca terrenos tan diversos como el educativo, el científico, el tecnológico o el social [11].

Cuando el objeto evaluado pertenece al campo de la ingeniería y del aprendizaje automático, esta lógica se traduce en medir hasta qué punto un modelo responde a los objetivos funcionales para los que fue diseñado. Aquí la evaluación deja de ser un trámite final y se convierte en el instrumento que permite comparar alternativas y decidir cuál conservar. En este trabajo esa comparación resulta ineludible, porque no se entrena un único clasificador sino cinco (la CNN base, la CNN con MobileNetV2, el MLP, el SVM y el Random Forest), todos apuntando a las mismas seis clases de expresión facial. Para distinguir su comportamiento se recurre a métricas específicas de clasificación, más allá de la simple exactitud, ya que un mismo porcentaje global puede esconder aciertos muy desiguales entre emociones. Ese análisis fue el que llevó, por ejemplo, a apartar al MLP de la aplicación en tiempo real: su exactitud de 86.67% quedó por debajo de la de los demás modelos, y solo un procedimiento de medición riguroso vuelve defendible una decisión de ese tipo.

Capítulo 3. Metodología

3.1 Tipo de Investigación

La investigación es de tipo aplicada, cuantitativa y experimental. Es aplicada porque se orienta al desarrollo de una solución tecnológica funcional basada en inteligencia artificial. Se considera cuantitativa porque utiliza métricas numéricas para evaluar el rendimiento de los modelos, y experimental debido a que el modelo fue entrenado, ajustado y probado bajo distintos parámetros para observar su comportamiento frente a datos previamente estructurados. En consecuencia, el trabajo se orienta por la siguiente hipótesis: en un escenario controlado y con un conjunto de datos reducido y homogéneo, los modelos de clasificación basados en características geométricas del rostro (SVM y Random Forest) pueden alcanzar un desempeño igual o superior al de las redes neuronales convolucionales, incluidas las que emplean Transfer Learning. De ella se derivan dos preguntas de investigación: (1) ¿qué arquitectura ofrece el mejor desempeño para el reconocimiento de expresiones faciales bajo condiciones controladas?, y (2) ¿el Transfer Learning con MobileNetV2 mejora el desempeño respecto a una CNN entrenada desde cero sobre un dataset pequeño?

3.2 Enfoque Metodológico General

El enfoque de esta investigación es de carácter cuantitativo, ya que se basa en la recolección, procesamiento y análisis de datos numéricos para evaluar el rendimiento del sistema de reconocimiento emocional.

El trabajo también sigue un enfoque tecnológico-computacional, al integrar técnicas de visión por computadora, redes neuronales convolucionales y bibliotecas de aprendizaje profundo como TensorFlow y MediaPipe para la construcción, entrenamiento y validación del sistema.

Este enfoque permite estructurar un proceso sistemático y replicable, orientado a resolver un problema práctico mediante la experimentación con herramientas avanzadas de inteligencia artificial.

3.3 Diseño Experimental

El presente trabajo adopta un diseño experimental de carácter cuantitativo, en el cual se desarrolló un sistema automático capaz de identificar emociones humanas a partir de imágenes faciales. El objetivo del experimento fue evaluar y comparar, en condiciones equivalentes de evaluación y sobre el mismo conjunto de datos, la efectividad de los cinco

modelos de clasificación implementados (CNN base, CNN con Transfer Learning basado en MobileNetV2, MLP, SVM y Random Forest), entrenados con imágenes previamente clasificadas siguiendo un protocolo estandarizado.

El proceso experimental constó de varias fases: recolección de imágenes faciales que representan estados emocionales (felicidad, tristeza, enojo, asombro, disgusto y neutralidad), preprocesamiento de dichas imágenes (conversión a escala de grises, redimensionamiento), detección de puntos clave mediante MediaPipe Face Mesh, y entrenamiento de los cinco modelos: las redes CNN base y MobileNetV2 junto con el MLP, implementados en TensorFlow y Keras, y el SVM y el Random Forest, implementados en scikit-learn.

Las variables controladas incluyen la iluminación, el tamaño de imagen (96×96 píxeles), el número de clases emocionales y el entorno de captura. Las variables dependientes fueron la exactitud de cada uno de los modelos, sus matrices de confusión y las métricas de clasificación por clase.

Este enfoque experimental permitió simular un entorno realista de reconocimiento emocional bajo condiciones reproducibles, lo que facilitó el análisis de rendimiento y la comparación entre los cinco modelos implementados.

3.4 Herramientas y Entornos

Para el desarrollo y evaluación del sistema de reconocimiento emocional, se utilizaron tanto recursos computacionales como equipamiento físico para la captura de datos en condiciones controladas.

La recolección de imágenes faciales se llevó a cabo en un entorno preparado específicamente para tal fin, utilizando un kit fotográfico profesional que incluía softboxes y sombrillas difusoras de luz para asegurar una iluminación homogénea y sin sombras duras (Figura 1). Las fotografías fueron tomadas con una cámara profesional Samsung NX2000, montada en un trípode para garantizar estabilidad y consistencia entre tomas (Figura 1).

Las capturas se realizaron en un piso del Centro de Ingeniería Avanzada (CIA) en la Facultad de Ingeniería, en una habitación abierta adaptada para trabajos. Se utilizó una pantalla verde (green screen) como fondo (Figura 2), aunque inicialmente se probó con fondo blanco (Figura 3). La elección del fondo verde se justificó por su eficacia en la posterior eliminación digital del fondo mediante un software de edición de imágenes (Figura 4), lo cual facilitó un mayor control en el preprocesamiento de imágenes y la homogeneidad de las muestras. Este entorno permite minimizar elementos distractores y mejorar la precisión de los algoritmos de detección facial.

Durante las sesiones de captura, el sujeto mantuvo una distancia y posición fija respecto a la cámara y se adoptaron diversas expresiones faciales correspondientes a emociones básicas

(felicidad, tristeza, enojo, entre otras), siguiendo un guion estandarizado para garantizar consistencia y facilitar la posterior clasificación automática de las emociones.

En cuanto al entorno computacional, el procesamiento de imágenes, la extracción de características faciales y el entrenamiento de modelos se realizaron en una laptop con procesador Intel Core i5 de 8ª generación, 16 GB de memoria RAM y tarjeta gráfica NVIDIA GeForce GTX 1060, operando sobre sistema Windows 10 con Python 3.10.

Las herramientas tecnológicas utilizadas fueron:

- MediaPipe Face Mesh: Para la detección y seguimiento de 468 puntos faciales en tiempo real.
- TensorFlow + Keras: Para el entrenamiento, validación y ejecución de los modelos de aprendizaje profundo (CNN base, CNN con MobileNetV2 y MLP) para clasificación emocional.
- scikit-learn: Para la implementación, entrenamiento y evaluación de los modelos de aprendizaje automático clásico (SVM y Random Forest), el escalamiento de características con StandardScaler y el cálculo de las métricas de desempeño.
- Python + OpenCV: Para la gestión de la cámara, procesamiento de imágenes, visualización y pruebas.
- Google Colab: Utilizado durante las fases iniciales para entrenamiento del modelo en la nube.
- Visual Studio Code: Entorno de desarrollo para la ejecución local del sistema integrado.

Figura 1. Kit fotográfico profesional

Figura 2. Kit fotográfico profesional con pantalla verde

Figura 3. Kit fotográfico profesional con pantalla blanca

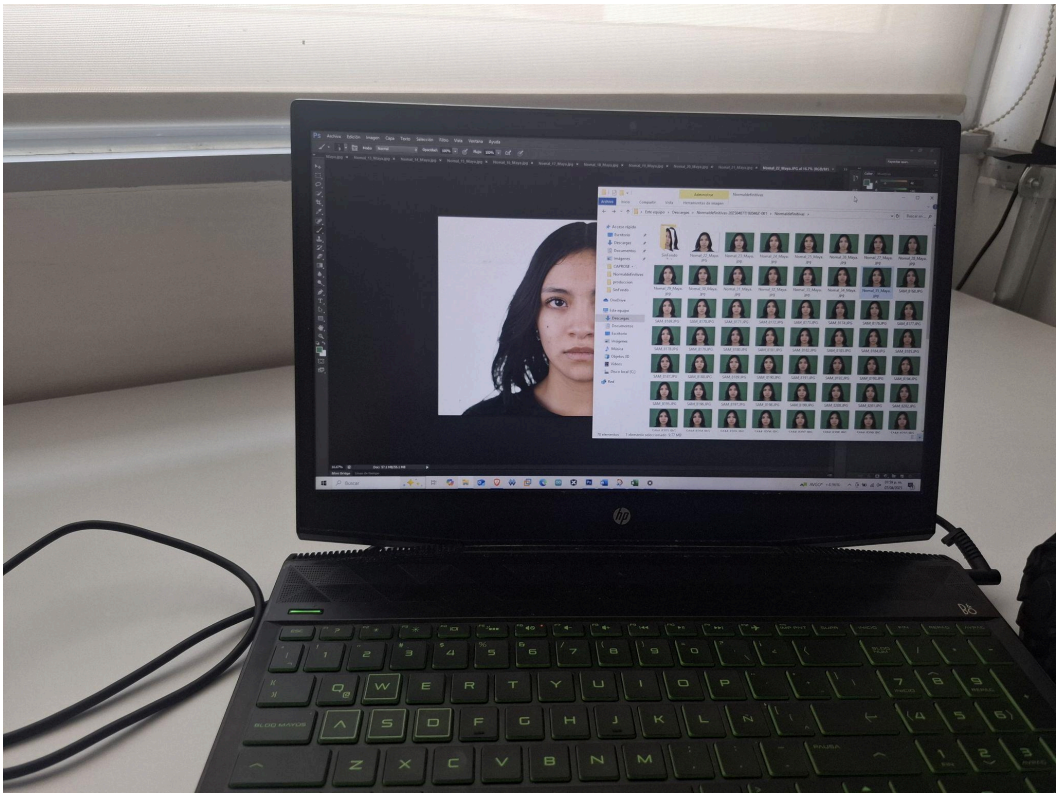

Figura 4. Edición de imágenes con software de edición de imágenes

3.5 Recolección y Preparación de Datos

La generación de la base de datos fue un componente fundamental del proyecto, ya que no se utilizó un dataset externo, sino que se creó uno personalizado con el objetivo de representar expresiones emocionales reales y controladas. La recolección de datos se llevó a cabo mediante sesiones fotográficas realizadas en un entorno controlado, específicamente en un piso del Centro de Ingeniería Avanzada (CIA) de la Facultad de Ingeniería de la UNAM.

Para garantizar la consistencia visual y minimizar fuentes de variabilidad, se implementó un protocolo de captura estandarizado:

Se utilizó un kit fotográfico profesional que incluía iluminación con softboxes y sombrillas, lo que permitió una exposición uniforme del rostro sin sombras marcadas.

El sujeto se ubicó a una distancia y altura fijas respecto a una cámara Samsung NX2000 montada sobre trípode, asegurando estabilidad y encuadres consistentes.

Como fondo se empleó una pantalla verde (green screen), lo cual facilitó la edición posterior y permitió aislar el rostro del entorno.

El participante fue instruido para representar estados emocionales (felicidad, tristeza, enojo, asombro, disgusto y neutralidad), siguiendo un guion y secuencia predeterminada.

Las imágenes fueron almacenadas en carpetas organizadas por emoción, asegurando un etiquetado preciso y coherente. Las carpetas se nombraron con las etiquetas "asombro", "disgusto", "enojo", "feliz", "normal" y "tristeza", que corresponden respectivamente a las emociones asombro, disgusto, enojo, felicidad, neutralidad y tristeza referidas en el texto. Estas mismas etiquetas abreviadas aparecen en la interfaz de la aplicación en tiempo real y en algunos pies de figura. Cada clase emocional se estructuró de forma balanceada para evitar sesgos durante el entrenamiento del modelo. Además, se llevaron registros visuales de las condiciones de captura, que incluyen ángulo, postura y expresión, con el fin de garantizar la validez de las muestras.

Este proceso riguroso de captura y organización sentó las bases para un conjunto de datos limpio, representativo y apto para tareas de aprendizaje supervisado, permitiendo posteriormente aplicar técnicas de preprocesamiento y entrenamiento con mayor eficiencia y confiabilidad.

La Figura 5 muestra ejemplos representativos de la base de datos construida, uno por cada emoción.

Figura 5. Ejemplos representativos de la base de datos para cada una de las seis emociones (asombro, disgusto, enojo, felicidad, neutralidad y tristeza)

3.6 Preprocesamiento de Datos

El preprocesamiento de imágenes fue una etapa esencial para garantizar la calidad, uniformidad y adecuación de los datos antes de ser utilizados en el entrenamiento del modelo de clasificación emocional. Esta fase permitió transformar las imágenes originales en un formato compatible con la arquitectura de red neuronal, reduciendo la variabilidad no deseada y optimizando el rendimiento del sistema.

Las principales operaciones realizadas fueron las siguientes:

- Conversión a escala de grises: se transformaron las imágenes RGB a escala de grises con el fin de reducir la complejidad computacional y eliminar la influencia de la información de color, la cual no es determinante para el reconocimiento de emociones faciales. Esto permitió simplificar el procesamiento y enfocar el análisis en la morfología facial (Figura 8).
- Redimensionamiento a 96×96 píxeles: todas las imágenes fueron redimensionadas a una resolución estándar de 96×96 píxeles. Esta estandarización facilita la entrada del modelo a una estructura fija y reduce la carga de memoria, manteniendo una

resolución suficiente para conservar detalles faciales relevantes para la detección de patrones emocionales (Figuras 8 y 9).

- Normalización de valores de píxel: los valores de intensidad de cada píxel fueron escalados al rango [0, 1], lo cual mejora la estabilidad numérica del entrenamiento. Este paso permite que el algoritmo de optimización converja más rápido al evitar que los valores altos dominen el cálculo del gradiente durante el aprendizaje.

- Eliminación del fondo: gracias al uso de una pantalla verde durante la captura (Figura 5), fue posible eliminar de forma precisa el fondo de las imágenes utilizando herramientas de edición como Adobe Photoshop CS6. Esta técnica reduce el ruido visual, enfocando la atención del sistema en las regiones faciales sin interferencias externas (Figura 7).

- Extracción de puntos faciales con MediaPipe: una vez preprocesadas las imágenes, se aplicó el modelo Face Mesh de MediaPipe para detectar automáticamente 468 puntos faciales (landmarks), los cuales capturan la geometría y estructura del rostro. Estos puntos se transformaron en vectores de entrada para alimentar modelos complementarios como SVM o redes neuronales.

- Aumento de datos (data augmentation): en algunos conjuntos de datos se aplicaron transformaciones artificiales como rotación, inversión horizontal y desplazamiento leve. Estas técnicas aumentan la diversidad de muestras y ayudan a prevenir el sobreajuste (overfitting), permitiendo que el modelo generalice mejor ante nuevas expresiones no vistas durante el entrenamiento.

En conjunto, estos pasos de preprocesamiento aseguraron que las imágenes alimentaran al modelo en un formato óptimo, homogéneo y robusto frente a la variabilidad natural de las expresiones faciales humanas.

Figura 6. Imagen original con fondo

Figura 7. Imagen original con color sin fondo

Figura 8. Imagen en escala de grises

Figura 9. Rostro recortado

Figura 10. Rostro en 96×96 píxeles

3.7 Detección Facial y Extracción de Características

La detección facial y la extracción automática de características se llevaron a cabo utilizando MediaPipe Face Mesh, una solución de visión por computadora desarrollada por Google, que permite el mapeo detallado del rostro humano a partir de una imagen en 2D. Esta herramienta es capaz de localizar hasta 468 puntos faciales (landmarks) (ver Figura 11) con alta precisión, sin necesidad de utilizar sensores de profundidad ni marcadores físicos.

Para cada imagen procesada, MediaPipe realiza en primer lugar una detección del rostro completo, delimitando su región dentro del encuadre. Posteriormente, sobre esta región detectada, se aplican algoritmos de regresión para calcular la posición exacta de cada uno de los puntos faciales. Estos landmarks abarcan áreas clave como:

- Ojos y párpados
- Cejas
- Nariz (puente, base y orificios nasales)

- Boca (labios superiores e inferiores)
- Mandíbula y contorno facial

Cada punto se expresa mediante una terna de coordenadas (x, y, z), relativas al ancho, alto y profundidad estimada de la imagen, lo que permite representar geométricamente la configuración tridimensional del rostro. Esta representación densa proporciona una descripción estructurada de las proporciones y la disposición facial de cada muestra.

Una vez extraídos, los datos de cada rostro fueron almacenados en archivos con formato .csv, donde cada fila corresponde a una imagen, y contiene la secuencia de coordenadas ordenadas. Este formato permitió organizar grandes cantidades de información de manera eficiente, facilitando el análisis posterior.

La extracción automatizada de landmarks faciales con MediaPipe ofreció una forma precisa, rápida y consistente de obtener características faciales relevantes. Este paso fue fundamental para representar cuantitativamente expresiones humanas con suficiente nivel de detalle para ser utilizadas en etapas de análisis y evaluación emocional.

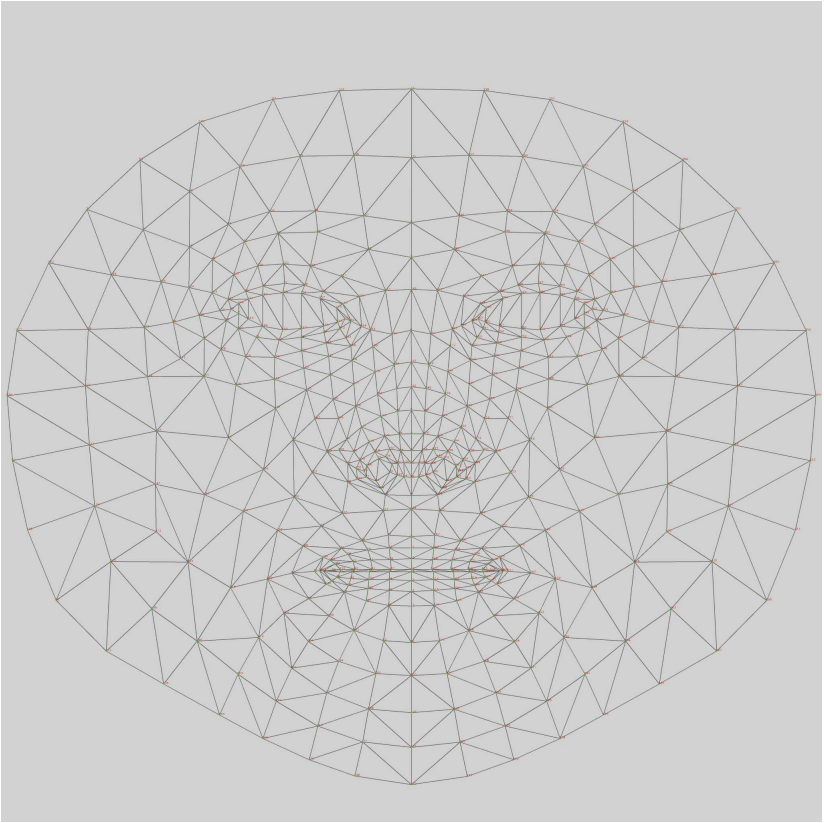

Figura 11. Distribución de los 468 puntos faciales utilizados por MediaPipe Face Mesh. Imagen recuperada de Stack Overflow [23]: https://stackoverflow.com/questions/69858216/mediapipe-facemesh-vertices-mapping

3.8 División de Datos (Entrenamiento/Prueba)

Para evaluar de forma objetiva el desempeño del modelo, el conjunto de imágenes recolectadas fue dividido en dos subconjuntos: entrenamiento y prueba. Esta división permitió separar el proceso de aprendizaje del modelo del proceso de evaluación, evitando sobreajuste y sesgos en las métricas finales.

Se utilizó una estrategia de partición aleatoria estratificada con las siguientes proporciones (en el caso del MLP se empleó una partición aleatoria simple, cuyo efecto es mínimo dado el balance del conjunto entre las seis clases):

- 80% del total de imágenes se destinaron al entrenamiento, permitiendo al modelo aprender los patrones asociados a cada emoción.
- El 20% restante fue reservado para el conjunto de prueba. Este subconjunto se empleó como conjunto de validación durante el entrenamiento —para el monitoreo de las curvas de aprendizaje, la detención temprana (EarlyStopping) y la selección del mejor modelo (ModelCheckpoint)— y, asimismo, para el reporte de las métricas finales. Debe señalarse que, al cumplir ambas funciones, los valores reportados corresponden al desempeño sobre el conjunto de validación y no sobre un conjunto de prueba estrictamente independiente.

Esta división se aplicó manteniendo el equilibrio entre clases emocionales en cada subconjunto. De esta manera, se garantiza una representación proporcional de todas las emociones durante las etapas de entrenamiento y evaluación, permitiendo una comparación justa entre los modelos bajo el mismo conjunto de datos controlado.

3.9 Arquitectura del Modelo CNN

El modelo de clasificación emocional propuesto en esta tesis se basa en una Red Neuronal Convolucional (CNN) implementada en TensorFlow y Keras. La arquitectura fue diseñada para procesar imágenes faciales preprocesadas de tamaño 96×96 píxeles en escala de grises, y tiene como objetivo identificar la emoción expresada en cada imagen entre varias clases posibles (por ejemplo: felicidad, tristeza, enojo, etc.).

La arquitectura está compuesta por una secuencia de capas especializadas, cuya organización se describe a continuación. En la Tabla 1 se muestra la arquitectura del modelo empleada:

Tabla 1. Arquitectura de la red neuronal convolucional implementada

Capa	Tipo	Descripción
1	Conv2D	32 filtros, tamaño de kernel 3×3, activación ReLU. Extrae características locales.
2	MaxPooling2D	Tamaño 2×2. Reduce la dimensionalidad espacial conservando información relevante.
3	Conv2D	64 filtros, kernel 3×3, activación ReLU. Capta patrones más complejos.
4	MaxPooling2D	Tamaño 2×2. Aplica más reducción espacial.
5	Conv2D	128 filtros, kernel 3×3, activación ReLU. Detecta combinaciones de patrones.
6	MaxPooling2D	Tamaño 2×2. Reduce la dimensionalidad antes de la capa densa.
7	Flatten	Convierte la salida 2D en un vector 1D para conectarse con la capa densa.
8	Dense	128 neuronas, activación ReLU. Capa totalmente conectada.
9	Dropout	Tasa 0.5. Previene sobreajuste desactivando aleatoriamente neuronas durante el entrenamiento.
10	Dense (output)	Número de neuronas igual al número de clases, activación Softmax para clasificación multiclase.

Detalles técnicos adicionales

- Entrada del modelo: Imagen de 96×96×1 (grises).
- Función de pérdida: *Categorical Crossentropy*, apropiada para clasificación multiclase.
- Optimizador: *Adam*, por su capacidad adaptativa y estabilidad en el aprendizaje.
- Métrica de evaluación: *Accuracy*.
- Épocas de entrenamiento: máximo de 25, con detención temprana (EarlyStopping)
- Tamaño de lote (batch size): 16

Esta configuración fue seleccionada por su equilibrio entre complejidad y eficiencia, permitiendo captar detalles faciales importantes sin incurrir en sobreajuste. La capa Dropout fue agregada para mejorar la generalización del modelo y reducir el riesgo de sobreentrenamiento.

3.10 Arquitectura del Modelo CNN Mejorado con Transfer Learning

Con el objetivo de mejorar la capacidad de generalización del sistema de reconocimiento emocional, se implementó una segunda arquitectura CNN basada en Transfer Learning, utilizando MobileNetV2 [26] como red base preentrenada. Este enfoque resulta especialmente adecuado en contextos donde el volumen de datos de entrenamiento es limitado, ya que aprovecha representaciones visuales aprendidas a partir de millones de imágenes para extraer características relevantes del rostro sin necesidad de entrenar la totalidad de la red desde cero [24].

A diferencia de la CNN base descrita anteriormente, este modelo recibe imágenes de 96×96 píxeles en formato RGB (tres canales de color), dado que MobileNetV2 fue preentrenada sobre el conjunto de datos ImageNet [27] utilizando imágenes a color. La arquitectura completa del modelo mejorado se compone de dos bloques diferenciados:

- ### 3.10.1 Red Base (MobileNetV2)

  Conformada por una arquitectura de 53 capas de profundidad organizadas en bloques residuales invertidos. Durante la primera fase de entrenamiento, la totalidad de estas capas permanece congelada, actuando únicamente como extractor de características. Las últimas 30 capas son descongeladas durante la segunda fase para el ajuste fino [25].

- ### 3.10.2 Cabeza Clasificadora Personalizada

  Bloque de capas densamente conectadas añadido sobre la red base, diseñado específicamente para la clasificación de las seis emociones del presente trabajo. Su estructura se detalla en la Tabla 2.

Tabla 2. Arquitectura de la cabeza clasificadora del modelo CNN mejorado (MobileNetV2)

Capa	Tipo	Descripción
1	GlobalAveragePooling2D	Reduce el mapa de características a un vector 1D
2	Dense	256 neuronas, activación ReLU
3	BatchNormalization	Normalización por lotes, estabiliza el entrenamiento
4	Dropout	Tasa 0.5. Regularización para prevenir sobreajuste
5	Dense	128 neuronas, activación ReLU
6	Dropout	Tasa 0.3
7	Dense (salida)	6 neuronas, activación Softmax — una por emoción

- Entrada del modelo: imagen de 96×96×3 (RGB).
- Función de pérdida: Categorical Crossentropy.
- Optimizador Fase 1: Adam con tasa de aprendizaje 1×10^{-3}.
- Optimizador Fase 2 (fine-tuning): Adam con tasa de aprendizaje 2×10^{-5}.

3.11 Estrategia de Entrenamiento en Dos Fases

El entrenamiento del modelo mejorado se estructuró en dos fases consecutivas siguiendo la metodología estándar de Transfer Learning [24]:

- ### 3.11.1 Fase 1: Extracción de Características (Feature Extraction)

  La red base MobileNetV2 se mantuvo completamente congelada. Únicamente la cabeza clasificadora fue entrenada durante un máximo de 35 épocas con batch size de 32.

  El callback EarlyStopping monitoreó la precisión de validación con una paciencia de 8 épocas, deteniendo el entrenamiento anticipadamente ante la ausencia de mejora.

  El callback ReduceLROnPlateau redujo la tasa de aprendizaje a la mitad cuando la pérdida de validación no mejoraba en 4 épocas consecutivas.

- ### 3.11.2 Fase 2: Ajuste Fino (Fine-tuning)

  Se descongelaron las últimas 30 capas de la red base MobileNetV2, manteniendo congeladas las capas inferiores que contienen características visuales de bajo nivel universales.

  El reentrenamiento se realizó durante un máximo de 25 épocas adicionales con una tasa de aprendizaje reducida (2×10^{-5}), permitiendo la adaptación gradual de las capas superiores de la red base a las características específicas del reconocimiento de expresiones faciales, sin comprometer el conocimiento previamente adquirido sobre ImageNet [27].

3.12 Aumento de Datos (Data Augmentation)

Para compensar el tamaño reducido del dataset de entrenamiento y mejorar la robustez del modelo ante variaciones no controladas en condiciones reales de uso, se aplicaron técnicas de aumento de datos durante el entrenamiento mediante la clase ImageDataGenerator de Keras [12]. Las transformaciones aplicadas se describen en la Tabla 3.

Tabla 3. Transformaciones de aumento de datos aplicadas durante el entrenamiento

Transformación	Rango/Valor	Propósito
Rotación	±15°	Variación de inclinación de cabeza
Desplazamiento horizontal	±10%	Variación de posición lateral
Desplazamiento vertical	±10%	Variación de posición vertical
Zoom	±15%	Variación de distancia a la cámara
Volteo horizontal	Activado	Simetría facial izquierda-derecha
Variación de brillo	[0.75, 1.25]	Robustez ante cambios de iluminación

Estas transformaciones se aplicaron exclusivamente al conjunto de entrenamiento; el conjunto de validación fue procesado únicamente con la normalización propia de MobileNetV2 (mobilenet_v2.preprocess_input, que reescala los píxeles al rango [-1, 1]), garantizando una evaluación objetiva del desempeño del modelo.

3.13 Evaluación del Sistema

Con el objetivo de validar el rendimiento del sistema de reconocimiento emocional desarrollado, se definió una estrategia de evaluación que combina enfoques cuantitativos y cualitativos. Esta sección describe los métodos y métricas que se emplearán para analizar la precisión y comportamiento del modelo, sin presentar aún los resultados obtenidos.

3.13.1 Evaluación Cuantitativa

El desempeño del modelo será medido utilizando métricas estándar para tareas de clasificación multiclase. Las métricas a aplicar incluyen:

- Precisión global (accuracy): porcentaje de predicciones correctas sobre el total de muestras del conjunto de prueba.

- Matriz de confusión: visualización del rendimiento por clase, permitiendo identificar aciertos y errores específicos en cada categoría emocional.
- Precisión, recall y F1-score por clase: métricas útiles para evaluar el rendimiento cuando existen clases desbalanceadas, proporcionando una visión detallada del comportamiento del modelo.
- Curvas de pérdida y precisión por época: utilizadas para monitorear el proceso de entrenamiento y detectar posibles problemas de sobreajuste.

Estas métricas serán calculadas utilizando herramientas como scikit-learn y matplotlib, y aplicadas sobre el subconjunto reservado (20%). Como se detalla en la sección 3.8, dicho subconjunto cumplió simultáneamente las funciones de validación durante el entrenamiento y de reporte de las métricas finales, lo que se declara de forma explícita para asegurar una evaluación transparente y replicable.

3.13.2 Evaluación Cualitativa

Además del análisis numérico, se realizará una revisión visual de los resultados generados por el sistema. Este enfoque cualitativo tiene como finalidad complementar la evaluación cuantitativa, observando el comportamiento del modelo en condiciones reales. Las acciones contempladas incluyen:

- Inspeccionar imágenes correctamente clasificadas para verificar la coherencia visual entre la expresión facial observada y la etiqueta asignada.
- Analizar casos de clasificación errónea, especialmente aquellos que involucren emociones similares, con el objetivo de identificar patrones de confusión.
- Detectar tendencias visuales, como mayor eficacia en expresiones faciales marcadas frente a una menor precisión en emociones sutiles o neutras.

Este análisis permitirá identificar fortalezas y limitaciones del sistema, así como establecer posibles líneas de mejora para futuras versiones del modelo.

Capítulo 4. Desarrollo del Sistema

4.1 Construcción e Integración

El sistema de reconocimiento emocional desarrollado en este trabajo se estructura como un pipeline modular compuesto por tres etapas principales: adquisición y preprocesamiento de la imagen facial, extracción de características y clasificación emocional, y generación de una respuesta adaptativa. Esta arquitectura modular permite que cada componente opere de forma independiente, facilitando la evaluación individual de los modelos y su eventual integración en plataformas robóticas.

El flujo general del sistema opera de la siguiente manera. En primer lugar, se captura una imagen facial mediante una cámara, la cual es sometida a las operaciones de preprocesamiento descritas en la sección de Metodología: eliminación de fondo, conversión a escala de grises (para la CNN base) o mantenimiento en formato RGB (para MobileNetV2), redimensionamiento a 96×96 píxeles y normalización de valores de píxel: al rango [0, 1] para la CNN base, y al rango [-1, 1] mediante mobilenet_v2.preprocess_input para MobileNetV2. A continuación, la imagen preprocesada es procesada por los modelos de clasificación, los cuales generan un vector de probabilidades asociado a cada una de las seis emociones. Finalmente, la emoción con mayor probabilidad es seleccionada como predicción, siempre que su valor de confianza supere el umbral mínimo establecido.

Se implementaron cinco modelos de clasificación organizados en dos enfoques complementarios. El enfoque basado en imágenes comprende la CNN base de tres capas convolucionales, que recibe imágenes en escala de grises de 96×96×1 píxeles, y la CNN mejorada con MobileNetV2 mediante Transfer Learning, que procesa imágenes en formato RGB de 96×96×3 píxeles. El enfoque basado en características geométricas comprende el perceptrón multicapa (MLP), que utiliza como entrada un vector de 1404 valores (los 468 landmarks faciales con sus tres coordenadas x, y, z), la Máquina de Vectores de Soporte (SVM) con kernel RBF y el modelo de Bosque Aleatorio (Random Forest), ambos alimentados con 11 características geométricas normalizadas. En todos los casos, la extracción de puntos faciales se realizó mediante MediaPipe Face Mesh.

Los cinco modelos fueron evaluados de forma independiente sobre el mismo conjunto de prueba, con el propósito de comparar objetivamente el desempeño de cada enfoque bajo condiciones equivalentes. Esta estrategia de evaluación comparativa permite identificar las fortalezas y limitaciones de cada arquitectura, proporcionando información valiosa para la selección del modelo más adecuado según los requerimientos específicos de una aplicación de interacción humano-robot.

4.2 Implementación de los modelos e integración del sistema

La implementación del sistema se llevó a cabo en Python 3.10, utilizando TensorFlow 2.x y Keras como frameworks principales para el desarrollo de los modelos basados en redes neuronales (CNN base, MobileNetV2 y MLP), y scikit-learn para los modelos de aprendizaje automático clásico (SVM y Random Forest).

Para la CNN base, el modelo fue construido mediante la API Sequential de Keras, siguiendo la arquitectura descrita en la Tabla 1. El entrenamiento se realizó durante un máximo de 25 épocas con EarlyStopping y un tamaño de lote de 16, utilizando el optimizador Adam y la función de pérdida Categorical Crossentropy. Las imágenes de entrada fueron preprocesadas mediante detección facial con Haar Cascade de OpenCV, seguida de conversión a escala de grises y redimensionamiento a 96×96 píxeles.

Para la CNN mejorada con MobileNetV2, se siguió la estrategia de entrenamiento en dos fases descrita en la sección de Metodología. La red base preentrenada sobre ImageNet fue importada mediante la clase tf.keras.applications.MobileNetV2 con el parámetro include_top=False, descartando las capas de clasificación originales y conservando únicamente el extractor de características. Sobre esta base se añadió la cabeza clasificadora personalizada detallada en la Tabla 2. Las imágenes fueron convertidas a formato RGB y se aplicaron las transformaciones de aumento de datos especificadas en la Tabla 3 mediante la clase ImageDataGenerator de Keras.

Para el MLP, se utilizó MediaPipe Face Mesh para extraer los 468 landmarks faciales tridimensionales de cada imagen. Las coordenadas (x, y, z) de estos puntos se aplanaron en un vector de 1404 valores, que fue normalizado mediante StandardScaler de scikit-learn y utilizado como entrada para una red de capas densamente conectadas (512→256→128 neuronas) con regularización L2 y Dropout, entrenada durante 15 épocas con un tamaño de lote de 32.

Para los modelos SVM y Random Forest, se extrajeron 11 características geométricas normalizadas a partir de los landmarks de MediaPipe Face Mesh: ancho de boca, alto de boca, ratio de sonrisa, apertura del ojo izquierdo, apertura del ojo derecho, apertura media de ojos, distancia ceja-ojo izquierdo, distancia ceja-ojo derecho, distancia entre cejas, asimetría facial y distancia nariz-boca. Estas características fueron calculadas como ratios normalizados respecto al ancho del rostro, lo que las hace invariantes a la escala de la imagen. Cabe señalar que la "apertura media de ojos" corresponde al promedio de las aperturas de los ojos izquierdo y derecho; se incluyó como una característica agregada complementaria, ya que los modelos empleados (Random Forest y SVM) son robustos a la presencia de características correlacionadas. El modelo SVM fue configurado con kernel RBF y parámetro de regularización C=10, mientras que el Random Forest fue implementado

con 300 árboles de decisión. Ambos modelos fueron entrenados y evaluados mediante scikit-learn.

Adicionalmente, se desarrolló una aplicación de demostración en tiempo real utilizando Flask como servidor web local (localhost:5000). Esta interfaz permite visualizar el flujo de la cámara en vivo con los puntos de MediaPipe Face Mesh superpuestos sobre el rostro, un cuadro delimitador (bounding box) con la emoción detectada, y un panel lateral que muestra las barras de probabilidad de CNN Base, MobileNetV2, SVM y RF de forma simultánea. El MLP, si bien fue entrenado y evaluado dentro del estudio comparativo (Capítulo 5), no se integró en la aplicación en tiempo real por haber obtenido el menor desempeño (86.67%); en su lugar se priorizaron los cuatro modelos de mayor exactitud, siendo el SVM particularmente adecuado para la ejecución en vivo por su eficiencia y estabilidad. Si bien esta aplicación no constituye un componente central de la evaluación formal de la tesis, sirve como herramienta de validación visual y demostración funcional del sistema en condiciones de uso real.

4.3 Lógica de Decisiones del Robot

Dado que el presente trabajo se enfoca en el desarrollo del módulo de percepción emocional como primer componente de un sistema de interacción humano-robot escalable, no se implementó un robot físico. No obstante, se definió una lógica de decisiones que establece las respuestas esperadas de un robot social ante cada estado emocional detectado, con el propósito de orientar futuras integraciones con plataformas robóticas.

El sistema de decisión se rige por un umbral de confianza mínimo de 0.70 (70%). Cuando el modelo de clasificación genera un vector de probabilidades para una imagen facial, se selecciona la emoción con el valor de confianza más alto. Si dicho valor es igual o superior al umbral de 0.70, la emoción se considera válida y se activa la respuesta correspondiente. En caso contrario, el sistema clasifica el estado emocional como "estado incierto" y no genera ninguna respuesta reactiva, evitando así comportamientos inadecuados derivados de predicciones poco confiables. Este mecanismo de filtrado es fundamental para garantizar la fiabilidad del sistema en escenarios reales de interacción, donde una clasificación errónea podría afectar negativamente la experiencia del usuario.

La Tabla 4 describe las respuestas adaptativas esperadas del robot social para cada una de las seis emociones reconocidas por el sistema.

Tabla 4. Respuestas adaptativas esperadas del robot social según la emoción detectada

Emoción detectada	**Confianza**	**Respuesta esperada del robot**
Felicidad	≥ 0.70	Reforzar la interacción positiva mediante respuestas empáticas, como expresiones de acompañamiento afectivo, comentarios motivacionales o continuación de la actividad en curso.
Tristeza	≥ 0.70	Adoptar un tono calmado y comprensivo, ofrecer palabras de apoyo emocional, reducir la velocidad de interacción y, de ser pertinente, sugerir una pausa o actividad reconfortante.
Enojo	≥ 0.70	Reducir la intensidad de los estímulos, evitar respuestas confrontativas, proporcionar espacio al usuario y, si es posible, redirigir la interacción hacia un tema o actividad neutral.
Disgusto	≥ 0.70	Identificar y suspender la actividad o estímulo que genera la reacción negativa, ofrecer una alternativa y ajustar el curso de la interacción.
Asombro	≥ 0.70	Mantener y potenciar la interacción actual, explorar la causa del asombro mediante preguntas o información adicional, y aprovechar el estado de atención elevada del usuario.

Neutralidad	≥ 0.70	Continuar con el flujo normal de la interacción sin modificaciones, manteniendo un comportamiento estable y predecible.
Estado incierto	< 0.70	No activar respuesta reactiva. El robot mantiene su comportamiento por defecto y espera una nueva lectura emocional con mayor nivel de confianza.

Esta tabla de decisiones constituye un marco de referencia para la integración del módulo de percepción emocional con sistemas robóticos. Su diseño se fundamenta en principios de robótica social que enfatizan la importancia de respuestas empáticas, adaptativas y contextualmente apropiadas para lograr interacciones humano-robot efectivas y aceptables por parte del usuario [3] [22]. La implementación física de estas respuestas —incluyendo la generación de gestos, expresiones faciales del robot, ajustes de tono de voz y selección de acciones— queda propuesta como línea de trabajo futuro.

Capítulo 5. Resultados

Esta sección presenta los resultados obtenidos durante el entrenamiento, validación y evaluación de los modelos de clasificación emocional implementados. Se incluyen métricas cuantitativas, análisis por clase, comparación entre arquitecturas y discusión de los hallazgos.

5.1 Resultados del Modelo CNN Base

El modelo CNN base, compuesto por tres capas convolucionales con arquitectura descrita en la Tabla 1, fue entrenado durante un máximo de 25 épocas con EarlyStopping y un tamaño de lote de 16 sobre el conjunto de imágenes faciales en escala de grises de 96×96 píxeles. La Figura 12 muestra la evolución de la precisión y la pérdida durante el entrenamiento.

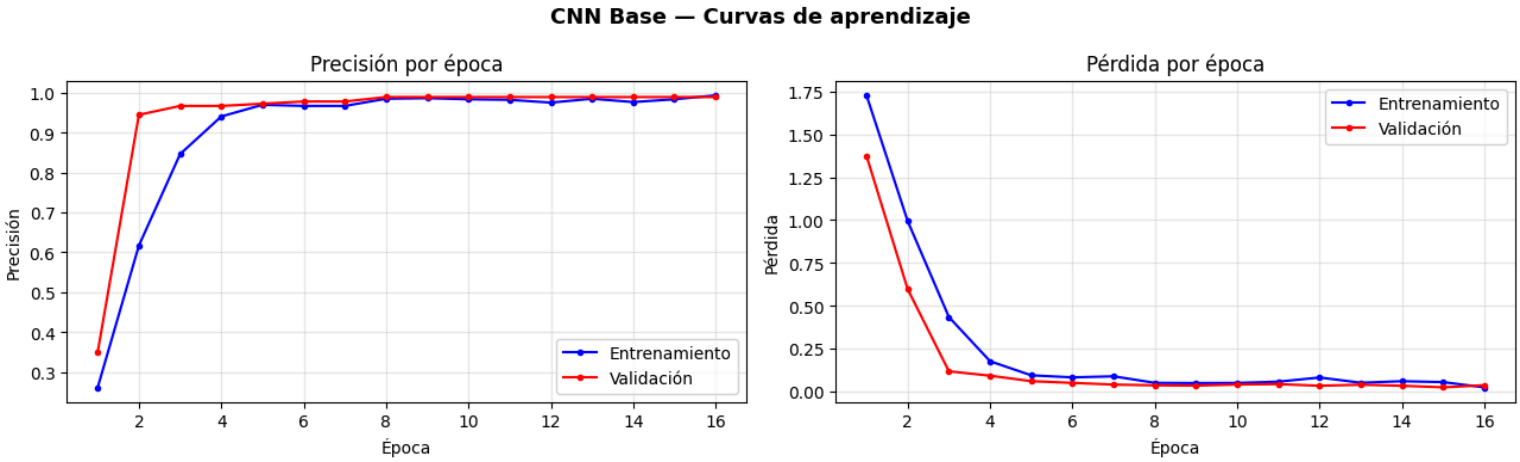

Figura 12. Curvas de accuracy y loss por época del modelo CNN Base

El modelo CNN base alcanzó una precisión global de 98.89% y un F1-score promedio de 0.9889 en el conjunto de prueba. Este resultado representa una mejora sustancial respecto a entrenamientos preliminares, atribuible a la optimización de los hiperparámetros y al preprocesamiento riguroso de las imágenes. Las curvas de entrenamiento muestran una convergencia estable, con una brecha reducida entre las curvas de entrenamiento y validación, lo que indica un nivel de sobreajuste mínimo. La Figura 13 presenta la matriz de confusión del modelo, y la Tabla 5 detalla las métricas de precisión, recall y F1-score por clase.

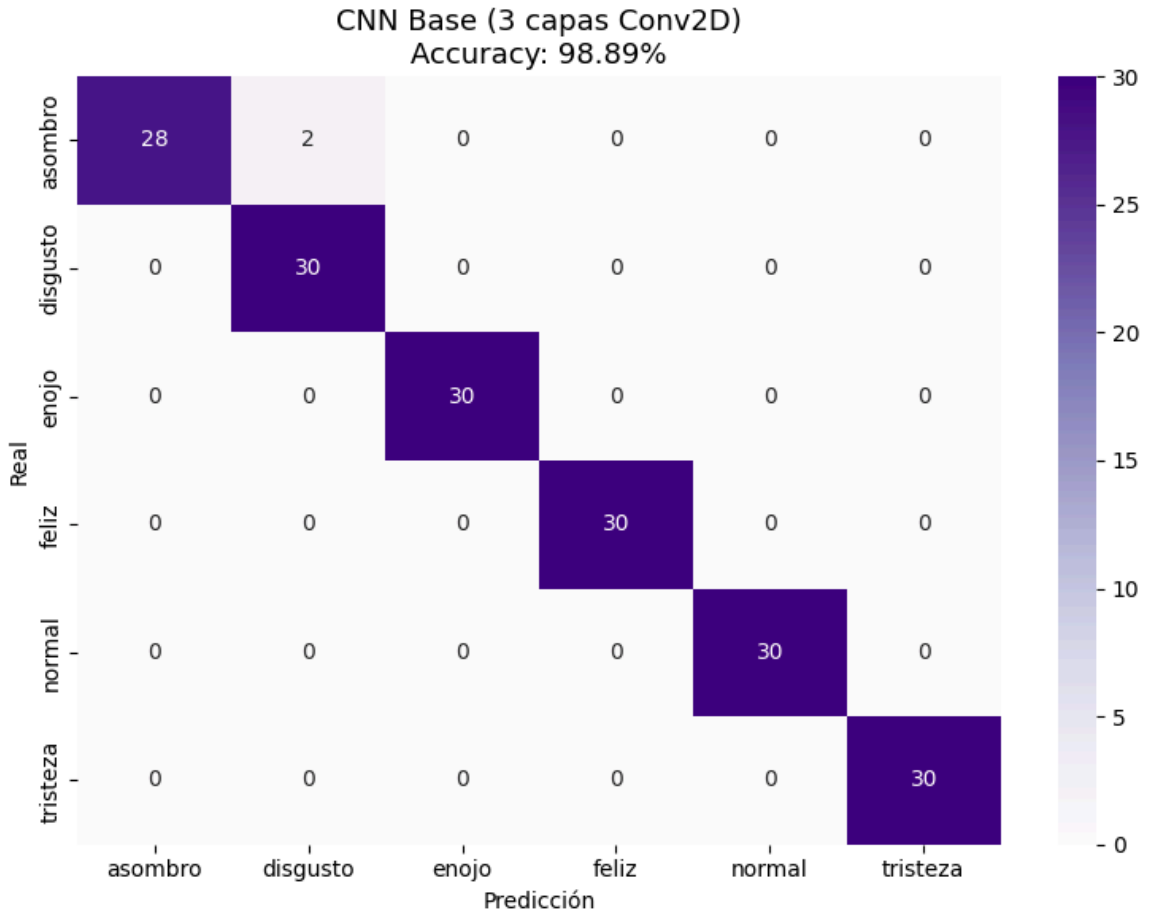

Figura 13. Matriz de confusión - CNN base

Tabla 5. Reporte de clasificación y métricas de desempeño del modelo CNN Base

📊 CNN Base — Accuracy: 0.9889 | F1-score: 0.9889

Reporte por clase:

	precision	recall	f1-score	support
asombro	1.00	0.93	0.97	30
disgusto	0.94	1.00	0.97	30
enojo	1.00	1.00	1.00	30
feliz	1.00	1.00	1.00	30
normal	1.00	1.00	1.00	30
tristeza	1.00	1.00	1.00	30
accuracy			0.99	180
macro avg	0.99	0.99	0.99	180
weighted avg	0.99	0.99	0.99	180

La alta precisión del modelo CNN base se explica por las condiciones controladas del dataset: imágenes capturadas con iluminación profesional uniforme, fondo eliminado digitalmente, un único sujeto de prueba y expresiones faciales bien diferenciadas. Estas condiciones minimizan la variabilidad que típicamente degrada el rendimiento de los modelos de reconocimiento emocional, validando la arquitectura propuesta como prueba de concepto funcional.

5.2 Resultados del Modelo CNN con MobileNetV2

El modelo basado en Transfer Learning con MobileNetV2 fue entrenado en dos fases siguiendo la metodología descrita en la sección de Metodología. Durante la Fase 1 (extracción de características), se entrenó únicamente la cabeza clasificadora con la red base completamente congelada. Durante la Fase 2 (ajuste fino), se descongelaron las últimas 30 capas de MobileNetV2 y se continuó el entrenamiento con una tasa de aprendizaje reducida de 2×10^{-5}. La Figura 14 muestra las curvas de precisión y pérdida para ambas fases, con una línea vertical indicando el inicio de la fase de ajuste fino.

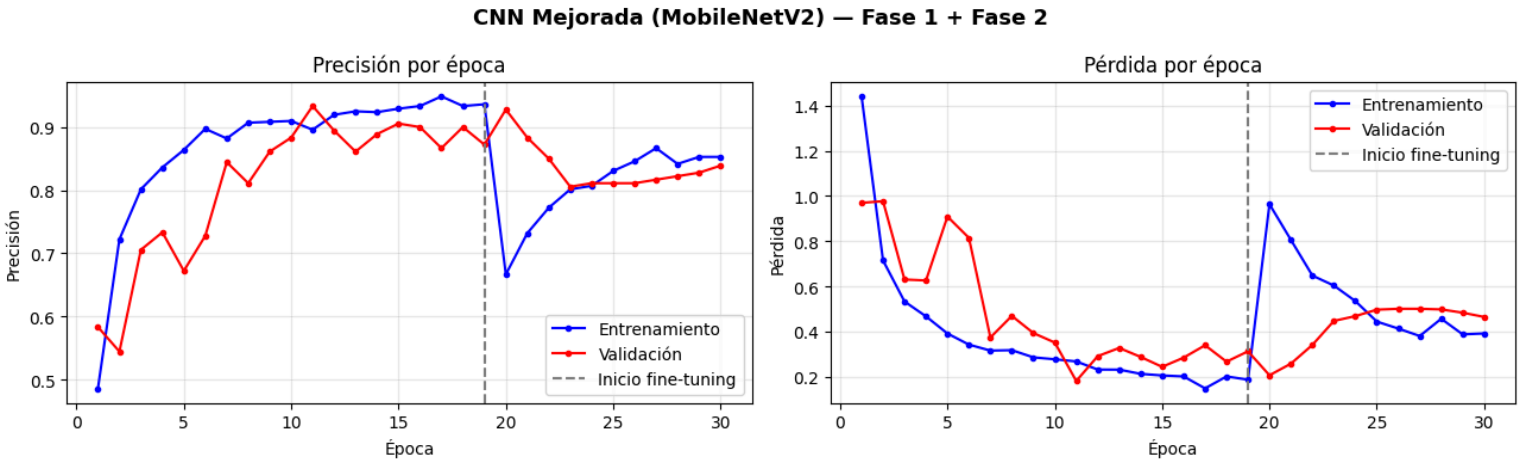

Figura 14. Curvas de accuracy y loss por época del modelo MobileNetV2 (Fase 1 + Fase 2, con línea vertical marcando inicio de fine-tuning)

El modelo MobileNetV2 alcanzó una precisión global de 92.78% y un F1-score promedio de 0.9288 en el conjunto de prueba. Si bien este resultado es inferior al obtenido por la CNN base (98.89%), este comportamiento es consistente con lo reportado en la literatura para escenarios donde el dataset de destino es significativamente más pequeño y menos diverso que el dataset de preentrenamiento. MobileNetV2 fue preentrenada sobre ImageNet (subconjunto ILSVRC-2012), un conjunto de aproximadamente 1.28 millones de imágenes distribuidas en 1,000 categorías [27], mientras que el dataset del presente trabajo contiene aproximadamente 900 imágenes de un único sujeto en 6 categorías. En estas condiciones, la complejidad de la arquitectura profunda de MobileNetV2 no ofrece una ventaja sobre la

CNN base más simple, que se adapta eficientemente a las características específicas del dataset reducido.

La Tabla 6 presenta las métricas detalladas por clase emocional para el modelo MobileNetV2, y la Figura 15 muestra la matriz de confusión correspondiente.

Tabla 6. Reporte de clasificación y métricas de desempeño del modelo MobileNetV2

Reporte por clase:

	precision	recall	f1-score	support
asombro	1.00	0.87	0.93	30
disgusto	0.80	0.93	0.86	30
enojo	0.83	0.83	0.83	30
feliz	0.97	1.00	0.98	30
normal	1.00	0.93	0.97	30
tristeza	1.00	1.00	1.00	30
accuracy			0.93	180
macro avg	0.93	0.93	0.93	180
weighted avg	0.93	0.93	0.93	180

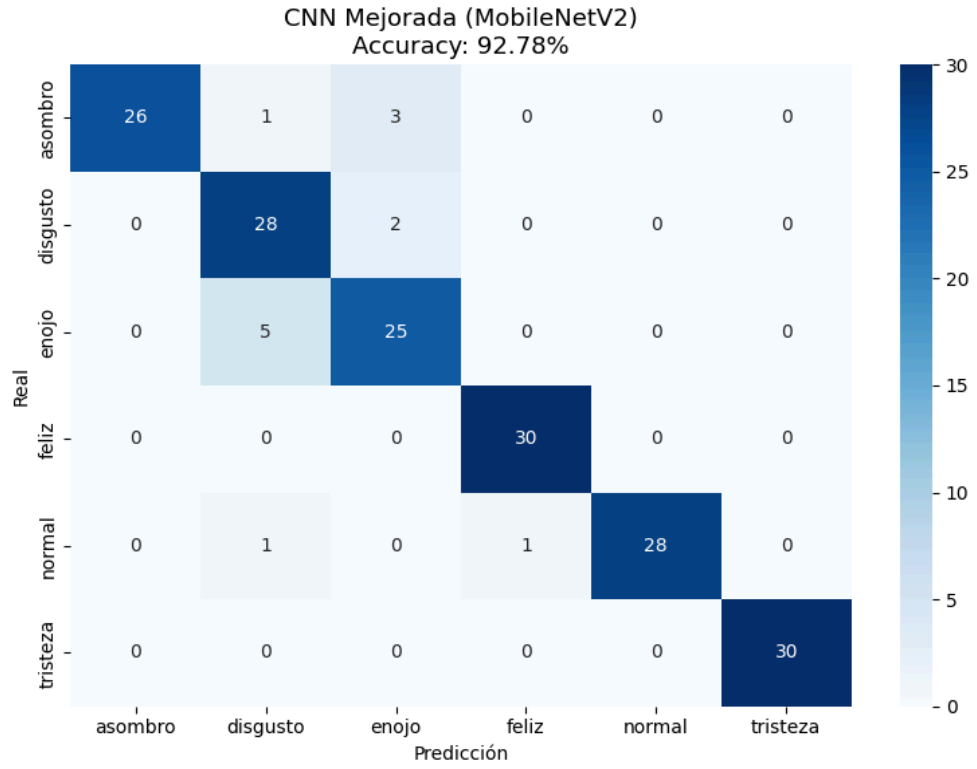

Figura 15. Matriz de confusión - MobileNetV2

El análisis por clase revela que el modelo MobileNetV2 alcanzó un desempeño perfecto en la clasificación de tristeza (F1 = 1.00) y resultados muy elevados en felicidad (F1 = 0.98) y neutralidad (F1 = 0.97). Las emociones con menor rendimiento fueron enojo (F1 = 0.83) y disgusto (F1 = 0.86), lo cual es consistente con la similitud de las configuraciones faciales entre estas emociones: ambas involucran tensión en la zona de la boca y el entrecejo, lo que dificulta su diferenciación incluso para modelos profundos. La emoción de asombro presentó una precisión perfecta (1.00) pero un recall de 0.87, indicando que el modelo fue muy preciso cuando predijo asombro, pero dejó de detectar un 13% de los casos reales, clasificándolos probablemente como otras emociones.

5.3 Resultados del Modelo MLP

El perceptrón multicapa, entrenado con los 1404 valores de los 468 puntos faciales tridimensionales (coordenadas x, y, z) extraídos mediante MediaPipe Face Mesh, fue desarrollado en una fase anterior del proyecto como modelo de referencia inicial. El MLP alcanzó una precisión global de 86.67% y un F1-score promedio de 0.8636 en el conjunto de prueba.

A diferencia de los modelos basados en imágenes, el MLP opera sobre un vector de características numéricas que representa la geometría del rostro. Este enfoque ofrece ventajas en términos de interpretabilidad, ya que permite analizar directamente qué proporciones faciales contribuyen a la clasificación de cada emoción. Sin embargo, la reducción de la información facial a un vector de landmarks implica una pérdida de información visual contextual —como texturas, sombras y microexpresiones— que las CNN sí son capaces de capturar, lo cual explica su menor rendimiento relativo. La Tabla 7 resume las métricas por clase del MLP, mientras que la Figura 16 muestra sus curvas de accuracy y loss durante el entrenamiento y la Figura 17 su matriz de confusión.

Tabla 7. Reporte de clasificación y métricas de desempeño del modelo MLP

MLP — Accuracy: 0.8667 | F1-score: 0.8636

Reporte por clase:

	precision	recall	f1-score	support
asombro	1.00	1.00	1.00	25
disgusto	0.89	0.57	0.70	28
enojo	0.69	0.85	0.76	34
feliz	1.00	1.00	1.00	32
normal	0.75	0.75	0.75	28
tristeza	0.94	1.00	0.97	33
accuracy			0.87	180
macro avg	0.88	0.86	0.86	180
weighted avg	0.87	0.87	0.86	180

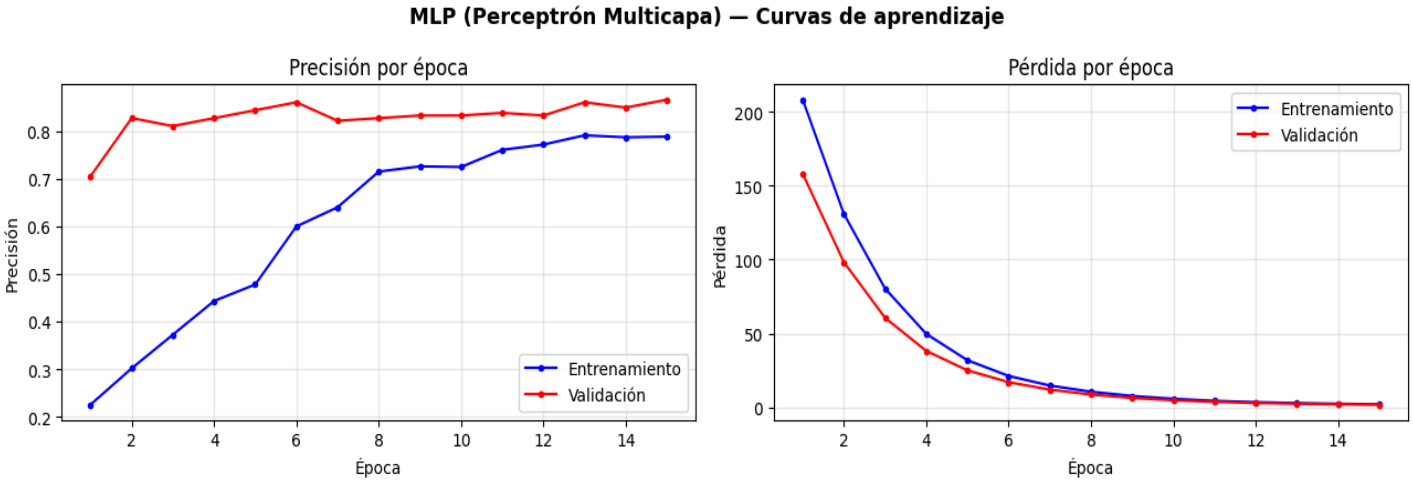

Figura 16. Curvas de accuracy y loss por época del modelo MLP

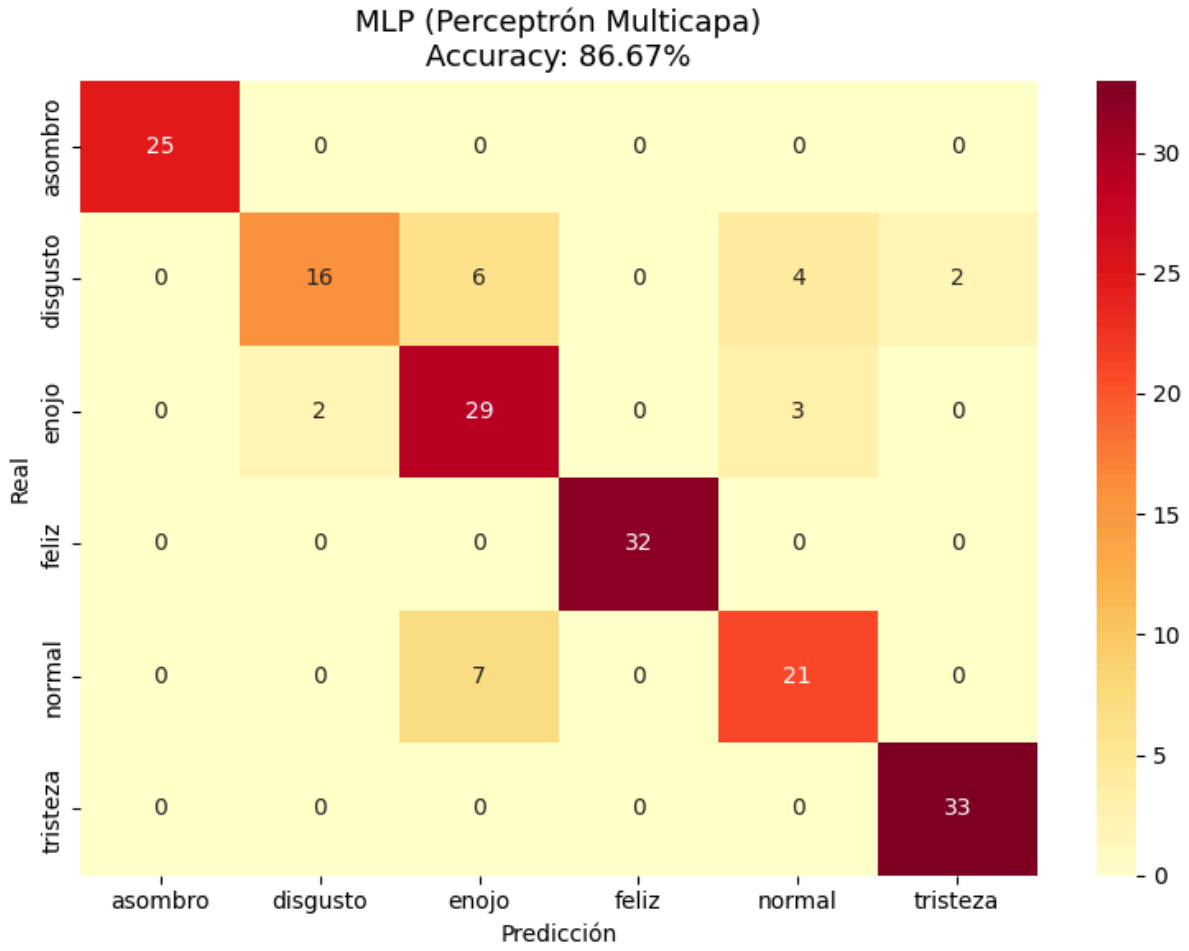

Figura 17. Matriz de confusión - MLP

5.4 Resultados de los Modelos SVM y Random Forest

Con el objetivo de evaluar el rendimiento de algoritmos de aprendizaje automático clásico sobre las mismas características geométricas del rostro, se implementaron un modelo de Máquina de Vectores de Soporte (SVM) con kernel RBF y parámetro de regularización C=10, y un modelo de Bosque Aleatorio (Random Forest) con 300 árboles de decisión. Ambos modelos fueron alimentados con 11 características geométricas normalizadas extraídas mediante MediaPipe Face Mesh: ancho de boca, alto de boca, ratio de sonrisa, apertura del ojo izquierdo, apertura del ojo derecho, apertura media de ojos, distancia ceja-ojo izquierdo, distancia ceja-ojo derecho, distancia entre cejas, asimetría facial y distancia nariz-boca.

El modelo SVM alcanzó una precisión global de 98.33% y un F1-score promedio de 0.9832 en el conjunto de prueba. La Figura 18 muestra su matriz de confusión y la Tabla 8 el reporte de métricas por clase.

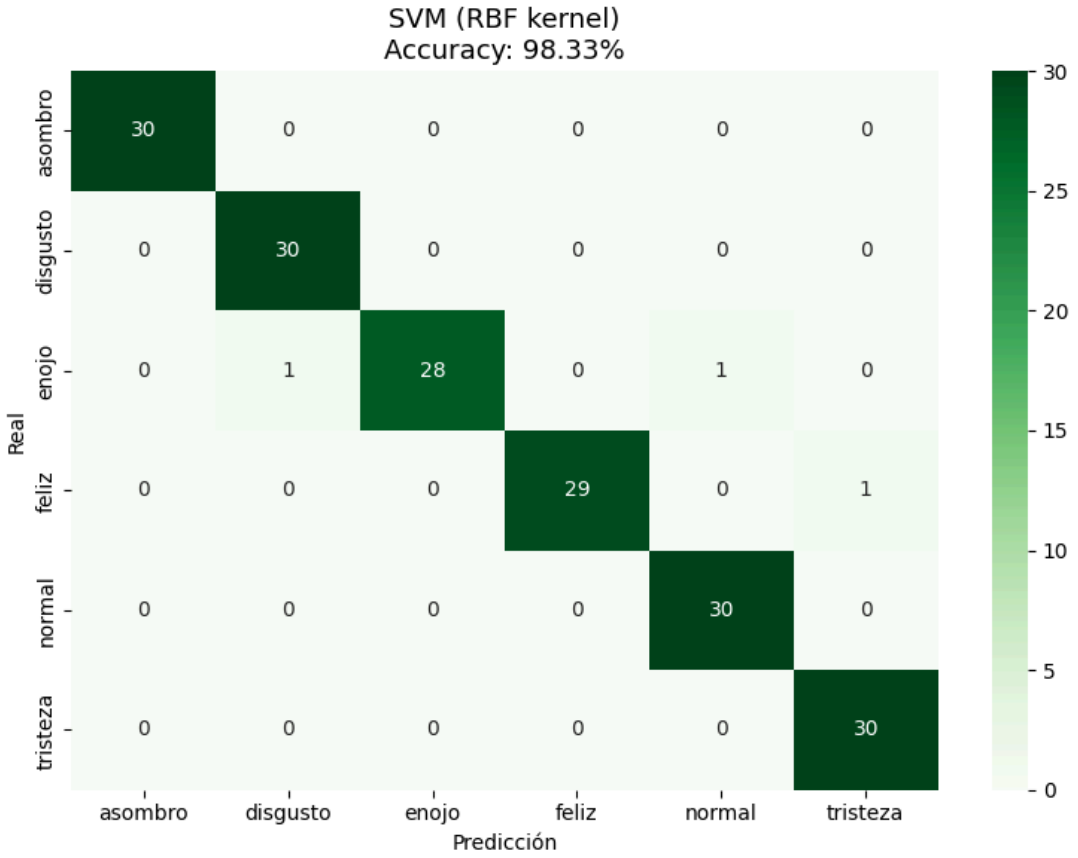

Figura 18. Matriz de confusión - SVM

Tabla 8. Reporte de clasificación y métricas de desempeño del modelo SVM

📊 SVM — Accuracy: 0.9833 | F1-score: 0.9832

	precision	recall	f1-score	support
asombro	1.00	1.00	1.00	30
disgusto	0.97	1.00	0.98	30
enojo	1.00	0.93	0.97	30
feliz	1.00	0.97	0.98	30
normal	0.97	1.00	0.98	30
tristeza	0.97	1.00	0.98	30
accuracy			0.98	180
macro avg	0.98	0.98	0.98	180
weighted avg	0.98	0.98	0.98	180

CV 5-fold: 0.9778 ± 0.0131

El modelo Random Forest alcanzó una precisión global de 99.44% y un F1-score promedio de 0.9944, constituyendo el modelo con mayor rendimiento entre los cinco evaluados. La Tabla 9 presenta sus métricas por clase; la Figura 19 muestra la importancia relativa de las 11 características geométricas en la decisión del modelo, y la Figura 20 su matriz de confusión.

Tabla 9. Reporte de clasificación y métricas de desempeño del modelo RF

📊 Random Forest — Accuracy: 0.9944 | F1-score: 0.9944

	precision	recall	f1-score	support
asombro	1.00	1.00	1.00	30
disgusto	1.00	0.97	0.98	30
enojo	1.00	1.00	1.00	30
feliz	1.00	1.00	1.00	30
normal	1.00	1.00	1.00	30
tristeza	0.97	1.00	0.98	30
accuracy			0.99	180
macro avg	0.99	0.99	0.99	180
weighted avg	0.99	0.99	0.99	180

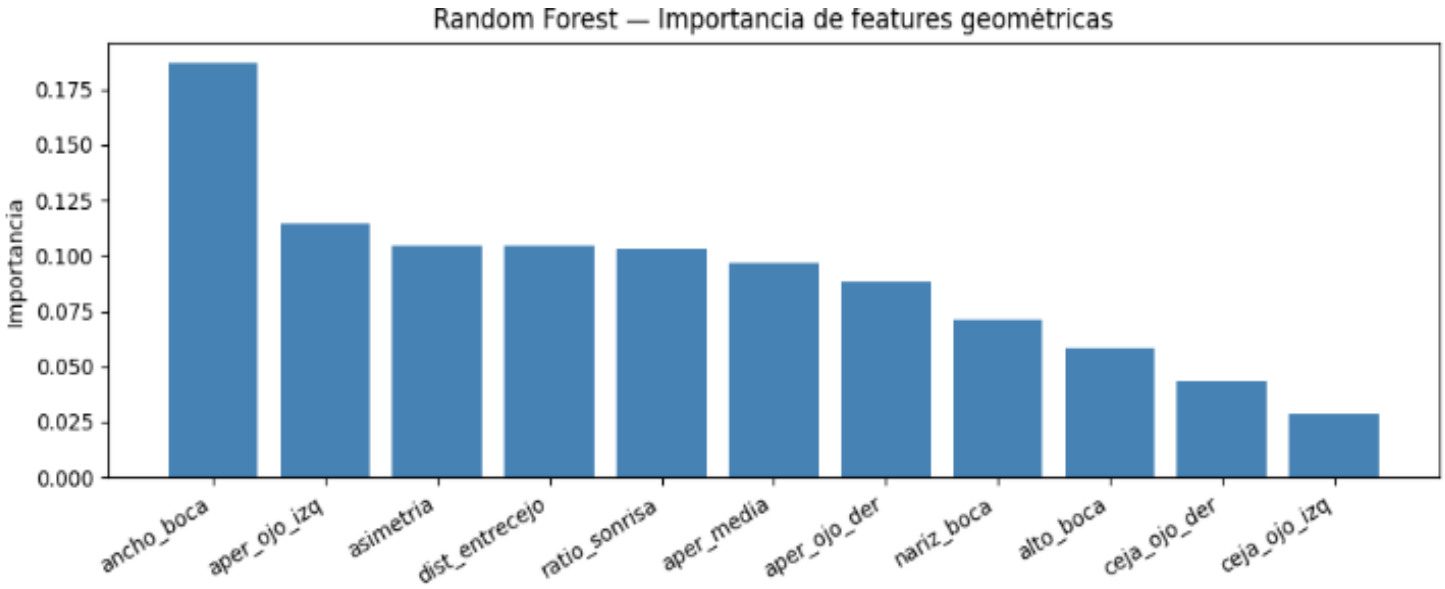

Figura 19. Importancia de las características faciales geométricas en el modelo Random Forest

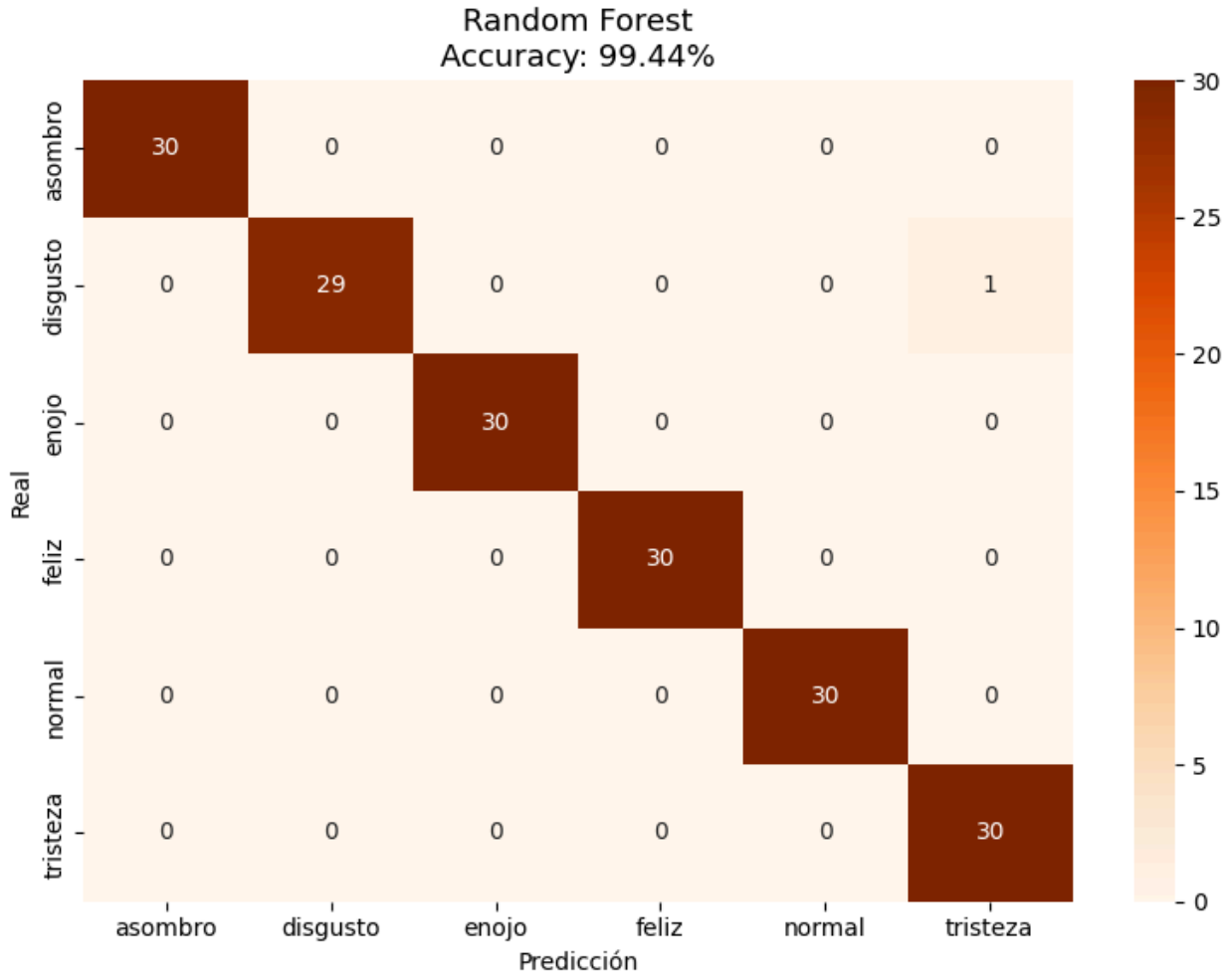

Figura 20. Matriz de confusión - Random Forest

El alto rendimiento de SVM y Random Forest sobre características geométricas resulta notable considerando la simplicidad de su entrada: únicamente 11 valores numéricos por imagen, frente a los 9,216 píxeles que procesa la CNN base o los 1404 valores del MLP. Este resultado sugiere que, en condiciones controladas con un único sujeto, las proporciones geométricas del rostro capturan de forma suficiente y eficiente la información discriminativa necesaria para la clasificación emocional. La normalización de estas características respecto al ancho del rostro las hace invariantes a la escala, lo que contribuye a la robustez de estos modelos.

5.5 Comparación entre Modelos

La Tabla 10 presenta una comparación global del rendimiento de los cinco modelos evaluados.

Tabla 10. Comparación de rendimiento entre los cinco modelos de clasificación emocional

Modelo	Entrada	Accuracy	F1-score
CNN Base (3 capas Conv2D)	Imagen 96×96 grises	98.89%	0.9889
CNN MobileNetV2 (Transfer Learning)	Imagen 96×96 RGB	92.78%	0.9288
MLP (Perceptrón multicapa)	1404 valores (468 landmarks × 3 coords)	86.67%	0.8636
SVM (kernel RBF, C=10)	11 features geométricas	98.33%	0.9832
Random Forest (300 árboles)	11 features geométricas	99.44%	0.9944

Como validación funcional del sistema integrado, las Figuras 21 y 22 muestran la clasificación en tiempo real de las emociones "feliz" y "asombro" mediante la cámara web, con las barras de probabilidad que los cuatro modelos integrados generan de forma simultánea en la aplicación.

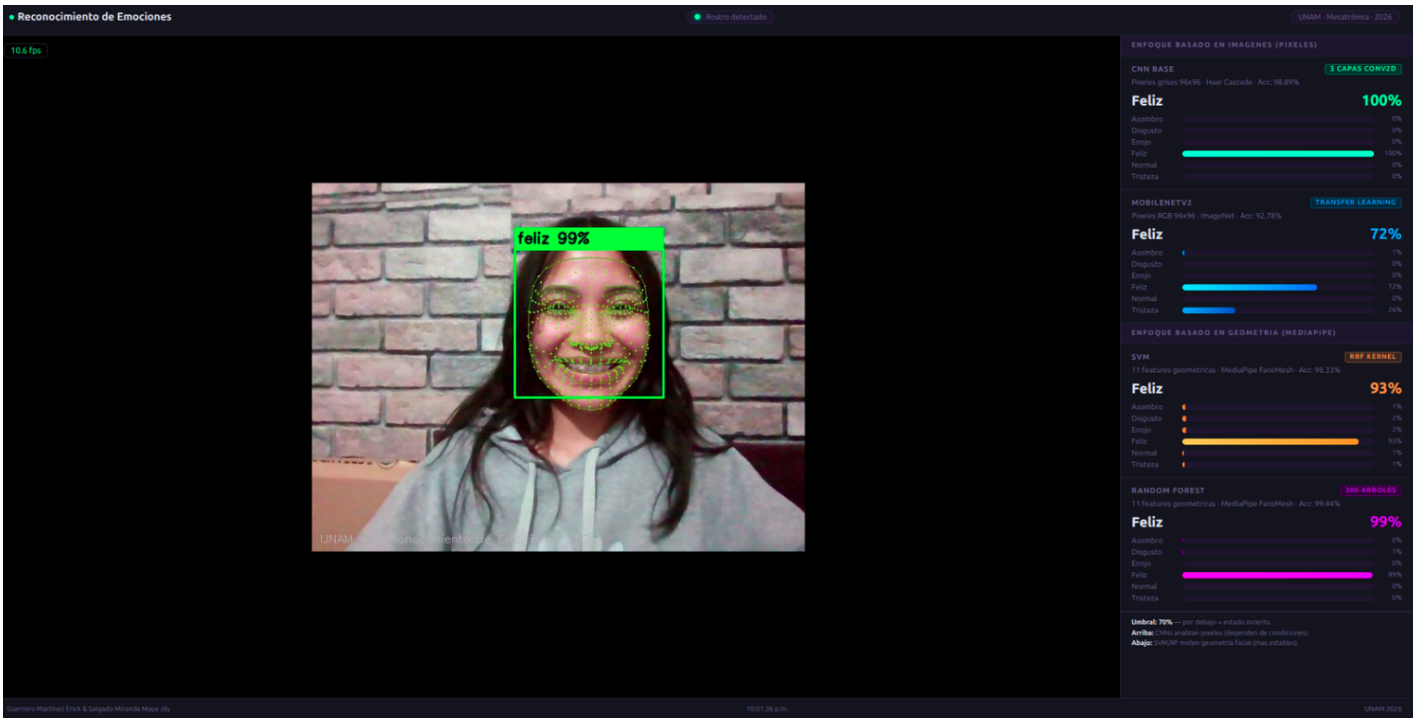

Figura 21. Clasificación de la emoción “feliz” mediante la cámara web y visualización de predicciones por modelo.

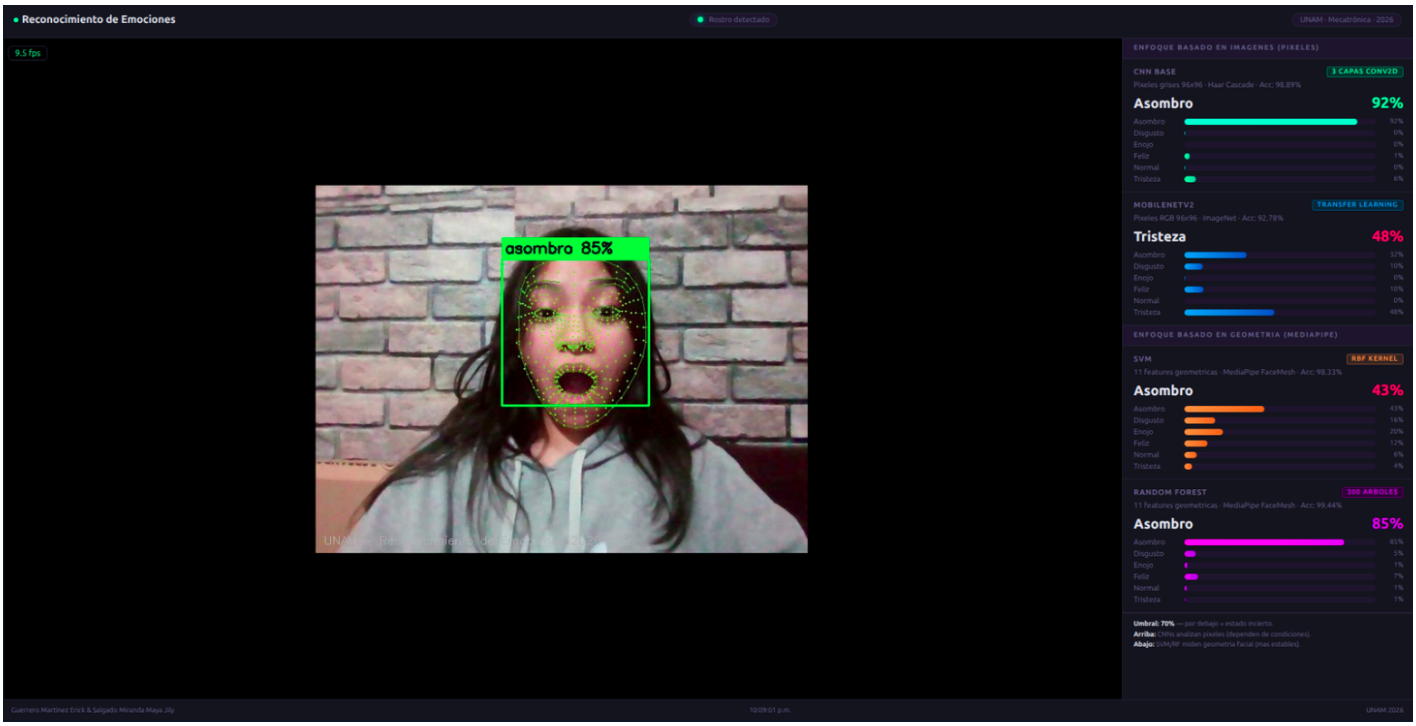

Figura 22. Clasificación de la emoción “asombro” mediante la cámara web y visualización de predicciones por modelo.

Los resultados permiten identificar varios hallazgos relevantes.

El modelo Random Forest obtuvo el mayor rendimiento global (99.44%), seguido de la CNN base (98.89%) y el SVM (98.33%). Estos tres modelos alcanzaron precisiones superiores al 98%, lo que demuestra la efectividad de los enfoques implementados para la tarea de clasificación emocional bajo las condiciones experimentales del presente trabajo.

Por su parte, el modelo MobileNetV2 con Transfer Learning (92.78%) no superó a la CNN base (98.89%) a pesar de contar con una arquitectura significativamente más profunda y compleja. Este resultado, lejos de invalidar el enfoque de Transfer Learning, ilustra una limitación conocida de esta técnica: cuando el dataset de destino es muy pequeño, muy homogéneo (un solo sujeto) y muy diferente del dataset de preentrenamiento (ImageNet contiene objetos generales, no expresiones faciales controladas), las representaciones preentrenadas pueden no transferirse de forma óptima al nuevo dominio [24] [25]. La CNN base, al ser entrenada desde cero sobre el dataset específico, pudo ajustar la totalidad de sus parámetros a las particularidades de las imágenes, lo que le otorgó ventaja en este escenario particular.

De manera destacada, el Random Forest superó a ambos modelos basados en imágenes, mientras que el SVM superó a MobileNetV2 y quedó marginalmente por debajo de la CNN base (98.33% frente a 98.89%). Este resultado se explica por la naturaleza del dataset: al tratarse de un único sujeto bajo condiciones controladas, las variaciones emocionales se manifiestan de forma consistente en proporciones geométricas específicas del rostro. Las 11 features seleccionadas capturan estas variaciones de forma compacta y eficiente, lo que permite que algoritmos clásicos con menor complejidad computacional alcancen rendimientos competitivos. Sin embargo, es importante señalar que esta ventaja podría reducirse significativamente en datasets con múltiples sujetos, condiciones de iluminación variables o expresiones de menor intensidad, donde las CNN suelen demostrar mayor capacidad de generalización.

Finalmente, el MLP obtuvo el rendimiento más bajo (86.67%). Esto se atribuye a la alta dimensionalidad de su entrada (468 landmarks × 3 coordenadas x, y, z = 1404 valores), que en combinación con un dataset de tamaño reducido, dificulta el aprendizaje de patrones generalizables. Los modelos SVM y Random Forest, que utilizan únicamente 11 features geométricas cuidadosamente seleccionadas, demostraron que una representación más compacta y discriminativa resulta más efectiva que una representación densa pero potencialmente redundante.

5.6 Discusión de Resultados

Es fundamental contextualizar los altos valores de precisión obtenidos (superiores al 98% en tres de los cinco modelos) dentro de las condiciones experimentales del presente trabajo. El dataset fue construido con un único sujeto de prueba, bajo condiciones controladas de iluminación, fondo y pose, con expresiones faciales de alta intensidad. Estas condiciones reducen significativamente la variabilidad inherente al problema de reconocimiento emocional, lo que favorece el rendimiento de los modelos. En datasets públicos de mayor diversidad, como FER-2013 o AffectNet, los modelos del estado del arte alcanzan precisiones considerablemente menores, típicamente entre 65% y 75% [10].

Por lo tanto, los resultados del presente trabajo no deben interpretarse como indicadores de rendimiento generalizable a escenarios no controlados, sino como una validación de la viabilidad técnica del sistema propuesto como prototipo funcional y prueba de concepto. La principal contribución radica en la evaluación comparativa de cinco enfoques bajo condiciones equivalentes, lo cual permite identificar las fortalezas y limitaciones de cada arquitectura y orientar el desarrollo futuro del sistema hacia escenarios más complejos y realistas.

Capítulo 6. Conclusiones

El presente trabajo desarrolló e implementó un sistema completo de reconocimiento automático de expresiones faciales basado en técnicas de aprendizaje automático y aprendizaje profundo, con aplicación directa en la mejora de la interacción humano-robot en entornos sociales. A lo largo del proyecto, se cumplieron los objetivos planteados y se obtuvieron resultados que validan la efectividad de la metodología propuesta.

6.1 Logros Principales

Se diseñaron, implementaron y evaluaron cinco modelos de reconocimiento emocional con enfoques complementarios. En el enfoque basado en imágenes, se desarrolló una CNN base de tres capas convolucionales que procesa imágenes de 96×96 píxeles en escala de grises, alcanzando una precisión de 98.89% en el conjunto de prueba; y una CNN mejorada basada en Transfer Learning con MobileNetV2 [26], preentrenada sobre ImageNet [27] y ajustada en dos fases con técnicas de aumento de datos, logrando una precisión de 92.78%. En el enfoque basado en características geométricas, se entrenaron un perceptrón multicapa (MLP) sobre 468 puntos faciales extraídos con MediaPipe Face Mesh (86.67% de precisión), una Máquina de Vectores de Soporte (SVM) con kernel RBF sobre 11 características geométricas normalizadas (98.33% de precisión), y un modelo de Bosque Aleatorio (Random Forest) con 300 árboles de decisión (99.44% de precisión). Contrastar los cinco modelos sobre el mismo conjunto de datos dejó en claro qué aporta y qué sacrifica cada arquitectura, un insumo directo para decidir cuál conviene desplegar en una aplicación de interacción humano-robot.

Se construyó una base de datos original de aproximadamente 900 imágenes faciales, capturadas bajo un protocolo estandarizado en un estudio fotográfico montado específicamente para este proyecto en las instalaciones del Centro de Ingeniería Avanzada (CIA) de la Facultad de Ingeniería, UNAM. Más allá del desarrollo técnico del sistema, esta etapa exigió aprender fotografía profesional, diseñar la iluminación del set, editar digitalmente las imágenes con software especializado y operar equipo de captura de alta calidad, una experiencia multidisciplinaria que amplió el alcance del proyecto.

El sistema desarrollado demostró capacidad para identificar seis estados emocionales (felicidad, tristeza, enojo, asombro, disgusto y neutralidad) con alto grado de precisión. Tres de los cinco modelos superaron el 98% de exactitud global, y el análisis por clase del modelo MobileNetV2 reveló un desempeño perfecto en la clasificación de tristeza (F1 = 1.00) y resultados superiores a 0.95 en felicidad y neutralidad. Los resultados obtenidos son consistentes con el comportamiento esperado para sistemas de reconocimiento emocional entrenados sobre datasets controlados con un único sujeto de prueba, y validan el sistema como prototipo funcional y prueba de concepto.

Se implementó un sistema modular, bien documentado y completamente funcional, que incluye desde la captura y preprocesamiento de imágenes hasta la predicción y visualización de resultados. La modularidad del código facilita su adaptación a diferentes contextos de uso, desde experimentación académica hasta integración en aplicaciones prácticas. Adicionalmente, se desarrolló una aplicación de demostración en tiempo real mediante Flask que permite visualizar el funcionamiento del sistema con cámara en vivo.

6.2 Aportaciones del Trabajo

Este proyecto contribuye al campo de la robótica social y la computación afectiva en varios aspectos.

Por un lado, aporta una implementación práctica completa y replicable que puede servir como base para futuros desarrollos en el área de reconocimiento emocional. El sistema abarca todas las etapas del pipeline, desde la adquisición de datos hasta la clasificación y la lógica de decisiones para la respuesta del robot.

Asimismo, la evaluación comparativa entre cinco modelos con enfoques fundamentalmente distintos —CNN base, CNN con Transfer Learning (MobileNetV2), MLP, SVM y Random Forest— constituye un análisis que permite identificar qué tipo de arquitectura resulta más adecuada según las características del dataset y los requerimientos de la aplicación. En particular, el hallazgo de que modelos clásicos como SVM y Random Forest pueden igualar o superar a redes neuronales profundas cuando el dataset es pequeño y controlado aporta evidencia relevante para la toma de decisiones en proyectos con restricciones similares.

También, la metodología de captura desarrollada, incluyendo el protocolo estandarizado de iluminación, pose y fondo, puede ser utilizada como referencia en trabajos futuros que requieran la creación de datasets controlados de expresiones faciales.

A ello se suma el análisis de características geométricas, que proporciona información valiosa sobre qué rasgos faciales diferencian cada emoción, mejorando la comprensión del problema y facilitando la validación de resultados desde una perspectiva interpretable.

Por último, el proyecto demuestra que el desarrollo de sistemas de inteligencia artificial aplicados a la robótica social requiere no solo conocimientos técnicos en programación y aprendizaje automático, sino también habilidades en áreas complementarias como fotografía, diseño experimental y procesamiento de señales visuales.

6.3 Limitaciones Identificadas

A pesar de los resultados positivos, el trabajo presenta limitaciones que deben ser consideradas.

La base de datos fue construida con un único sujeto de prueba, lo que limita la capacidad del sistema para generalizar a rostros con características faciales diferentes, como variaciones en estructura ósea, forma de ojos, textura de piel y rasgos demográficos. Esta limitación explica en gran medida los altos valores de precisión obtenidos, ya que el modelo aprende las particularidades de un solo rostro en lugar de patrones faciales universales.

El sistema fue entrenado con imágenes capturadas bajo condiciones controladas de iluminación y fondo, lo que puede afectar su rendimiento en escenarios reales con iluminación natural variable, fondos complejos o distancias de captura diferentes a las del entrenamiento.

El modelo MobileNetV2, a pesar de ser la arquitectura más sofisticada implementada, obtuvo un rendimiento inferior al de la CNN base y los modelos de características geométricas. Esto evidencia que el Transfer Learning desde ImageNet no necesariamente mejora el desempeño cuando el dataset de destino es muy pequeño, homogéneo y especializado, y que la selección de la arquitectura debe considerar las características específicas del problema y los datos disponibles.

Las emociones con configuraciones faciales similares, como enojo y disgusto, presentaron mayor dificultad de clasificación en el modelo MobileNetV2, lo que sugiere la necesidad de explorar representaciones adicionales que capturen diferencias más sutiles entre estas categorías emocionales.

Las restricciones temporales del proyecto limitaron la cantidad de datos recolectados y la exploración de técnicas más avanzadas de aumento de datos y arquitecturas adicionales.

Estas limitaciones no invalidan los resultados obtenidos, sino que establecen el alcance del trabajo como prototipo funcional y proporcionan direcciones claras para mejoras futuras.

6.4 Trabajo Futuro

Con base en los resultados obtenidos y las limitaciones identificadas, se proponen las siguientes líneas de investigación y desarrollo futuro.

Una primera línea de trabajo es ampliar el dataset incorporando múltiples sujetos con diferentes características demográficas (edad, género, etnia) para mejorar la capacidad de generalización del sistema. Se recomienda capturar al menos 50 a 100 sujetos con 30 a 50

imágenes por emoción cada uno, lo cual permitiría evaluar el rendimiento real de los modelos en condiciones de mayor variabilidad.

Otra vía prometedora son los modelos generativos, como Redes Generativas Adversariales (GANs) o modelos de difusión, para la síntesis de imágenes faciales que amplíen la diversidad del dataset más allá de las transformaciones geométricas convencionales implementadas en el presente trabajo, incorporando variaciones de características faciales entre sujetos distintos.

En cuanto a arquitecturas, convendría evaluar Transfer Learning más recientes, como EfficientNetV2 y Vision Transformer (ViT), comparando su desempeño con el modelo MobileNetV2 implementado. Estas arquitecturas han demostrado mejoras significativas en tareas de clasificación de imágenes y podrían ofrecer una mejor transferencia de conocimiento al dominio del reconocimiento emocional facial.

Se recomienda implementar un sistema de fusión de modelos (ensemble) que combine las predicciones de los clasificadores mediante votación ponderada o meta-aprendizaje, aprovechando las fortalezas complementarias de los enfoques basados en imágenes y en características geométricas.

Se sugiere incorporar redes recurrentes (LSTM) o arquitecturas basadas en transformadores para analizar secuencias temporales de expresiones faciales, permitiendo detectar transiciones emocionales y microexpresiones que un análisis de imagen estática no puede capturar.

Se propone el despliegue del sistema en una plataforma robótica real para evaluar su desempeño en escenarios auténticos de interacción humano-robot, implementando las respuestas adaptativas propuestas en la Tabla 4 y midiendo métricas como tiempo de respuesta, satisfacción del usuario y naturalidad de la interacción.

Se recomienda realizar estudios de validación con usuarios externos e independientes para evaluar la percepción, aceptación y utilidad del sistema en aplicaciones reales.

Finalmente, se propone explorar técnicas de compresión de modelos, como cuantización, poda (pruning) y destilación de conocimiento (knowledge distillation), para permitir la ejecución eficiente del sistema en dispositivos embebidos con recursos limitados, como Raspberry Pi o Jetson Nano.

6.5 Reflexiones Finales

El desarrollo de este proyecto representó un reto técnico y personal significativo, que permitió integrar conocimientos de múltiples áreas de la ingeniería mecatrónica: inteligencia artificial, visión por computadora, procesamiento de señales, diseño experimental y desarrollo de software. La experiencia adquirida en la construcción del estudio fotográfico, la captura sistemática de datos y la implementación de diversas arquitecturas de aprendizaje automático y aprendizaje profundo proporciona una base sólida para futuros proyectos en el campo de la robótica social y la computación afectiva.

El sistema desarrollado constituye una prueba de concepto funcional que demuestra la viabilidad técnica del reconocimiento automático de emociones faciales mediante técnicas de inteligencia artificial. La evaluación comparativa de cinco modelos con enfoques distintos permitió obtener una visión integral del problema, evidenciando que la selección del modelo óptimo depende de factores como el tamaño y diversidad del dataset, los recursos computacionales disponibles y los requerimientos específicos de la aplicación.

La incorporación de Transfer Learning mediante MobileNetV2, combinada con técnicas de aumento de datos y una estrategia de entrenamiento en dos fases, demostró que si bien esta técnica es valiosa en el estado del arte, su efectividad depende críticamente de la compatibilidad entre el dominio de preentrenamiento y el dominio de destino [24]. Al mismo tiempo, el alto rendimiento alcanzado por los modelos SVM y Random Forest con únicamente 11 características geométricas demuestra que, en escenarios controlados, representaciones compactas e interpretables pueden ser tan efectivas o más que arquitecturas profundas, un hallazgo relevante para aplicaciones en dispositivos con recursos computacionales limitados.

En conclusión, este trabajo aporta al campo de la interacción humano-robot un sistema experimental cuya metodología de captura, preprocesamiento y evaluación comparativa queda documentada de forma que pueda reproducirse y ampliarse en investigaciones futuras en el área de reconocimiento emocional y computación afectiva aplicada a la robótica social.

Referencias Bibliográficas

[1] S. Russell y P. Norvig, *Artificial Intelligence: A Modern Approach*, 4a ed. Upper Saddle River, NJ, EE. UU.: Pearson, 2021.

[2] R. Szeliski, *Computer Vision: Algorithms and Applications*. Londres, Reino Unido: Springer, 2011.

[3] T. Fong, I. Nourbakhsh y K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 143-166, 2003.

[4] M. A. Goodrich y A. C. Schultz, "Human-robot interaction: A survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, no. 3, pp. 203-275, 2007. doi: 10.1561/1100000005

[5] R. A. Calvo y S. D'Mello, "Affect detection: An interdisciplinary review of models, methods, and their applications," *IEEE Trans. Affective Comput*., vol. 1, no. 1, pp. 18-37, 2010.

[6] P. Ekman y W. V. Friesen, *Facial Action Coding System (FACS): Manual*. Palo Alto, CA, EE. UU.: Consulting Psychologists Press, 1978.

[7] Z. Zhang, P. Luo, C. C. Loy y X. Tang, "Facial landmark detection by deep multi-task learning," en *Proc. European Conf. Computer Vision (ECCV)*, Zúrich, Suiza, 2014, pp. 94-108.

[8] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. Cambridge, MA, EE. UU.: MIT Press, 2016. [En línea]. Disponible en: https://www.deeplearningbook.org/

[9] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2a ed. Sebastopol, CA, EE. UU.: O'Reilly Media, 2019.

[10] S. Li y W. Deng, "Deep facial expression recognition: A survey," *IEEE Trans. Affective Comput*., vol. 13, no. 3, pp. 1195-1215, 2022. doi: 10.1109/TAFFC.2020.2981446

[11] D. L. Stufflebeam y C. L. S. Coryn, *Evaluation Theory, Models, and Applications*, 2a ed. San Francisco, CA, EE. UU.: Jossey-Bass, 2014.

[12] F. Chollet, *Deep Learning with Python*, 1a ed. Shelter Island, NY, EE. UU.: Manning Publications, 2018.

[13] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg y M. Grundmann, "MediaPipe: A framework for building perception pipelines," arXiv preprint arXiv:1906.08172, 2019. [En línea]. Disponible en: https://arxiv.org/abs/1906.08172

[14] Google Developers, "MediaPipe Face Mesh," s.f. [En línea]. Disponible en: https://google.github.io/mediapipe/solutions/face_mesh

[15] R. C. Gonzalez y R. E. Woods, *Digital Image Processing*, 4a ed. Nueva York, NY, EE. UU.: Pearson, 2018.

[16] E. H. Houssein, A. E. Hassanien y D. Oliva, "Deep learning and machine learning models for emotion recognition from human facial expressions: A comprehensive review," *Expert Systems with Applications*, vol. 213, art. 119153, 2023.

[17] R. Elmasri y S. B. Navathe, *Fundamentals of Database Systems*, 7a ed. Boston, MA, EE. UU.: Pearson, 2017.

[18] A. K. Jain, A. Ross y K. Nandakumar, *Introduction to Biometrics*. Nueva York, NY, EE. UU.: Springer, 2011.

[19] T. M. Mitchell, *Machine Learning*. Nueva York, NY, EE. UU.: McGraw-Hill, 1997.

[20] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham, Suiza: Springer, 2018.

[21] D. Poole y A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, 2a ed. Cambridge, Reino Unido: Cambridge University Press, 2017. [En línea]. Disponible en: https://artint.info/2e/html/ArtInt2e.html

[22] K. Dautenhahn, "Socially intelligent robots: Dimensions of human-robot interaction," *Philosophical Transactions of the Royal Society B*, vol. 362, no. 1480, pp. 679-704, 2007. doi: 10.1098/rstb.2006.2004

[23] A. Bhargava, "Mediapipe facemesh vertices mapping," *Stack Overflow*, nov. 2021. [En línea]. Disponible en: https://stackoverflow.com/questions/69858216/mediapipe-facemesh-vertices-mapping

[24] S. J. Pan y Q. Yang, "A survey on transfer learning," *IEEE Trans*. Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, oct. 2010. doi: 10.1109/TKDE.2009.191

[25] J. Yosinski, J. Clune, Y. Bengio y H. Lipson, "How transferable are features in deep neural networks?" en *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014, pp. 3320-3328.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov y L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," en *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510-4520. doi: 10.1109/CVPR.2018.00474

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li y L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," en *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248-255. doi: 10.1109/CVPR.2009.5206848

[28] F. Chollet et al., "Keras," GitHub, 2015. [En línea]. Disponible en: https://keras.io

[29] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto y H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint, arXiv:1704.04861*, 2017. [En línea]. Disponible en: https://arxiv.org/abs/1704.04861

Anexos

Anexo A — Código fuente del notebook de Google Colab

```
Original file is located at
    https://colab.research.google.com/drive/1IjcsyXw1xFG7fJdnKXbgU6GbPcNq2zyv

# Sistema de Reconocimiento de Emociones Faciales
## Tesis — Notebook Final

**Autores:** Guerrero Martínez Erick | Salgado Miranda Maya Jily
**UNAM — Facultad de Ingeniería — Ingeniería Mecatrónica**

### Modelos implementados:
| # | Modelo | Entrada | Descripción |
|---|--------|---------|-------------|
| 1A | **CNN Base** | 96×96 grises | Arquitectura original de la tesis (3 capas Conv2D) |
| 1B | **CNN Mejorada** | 96×96 RGB | Transfer Learning con MobileNetV2 |
| 2 | **SVM** | 11 features geométricas | MediaPipe FaceMesh + kernel RBF |
| 3 | **Random Forest** | 11 features geométricas | 300 árboles, class_weight balanced |
| 4 | **MLP** |1404 valores (468 landmarks × 3) (MediaPipe) | Perceptrón multicapa 512→256→128 |

### Emociones: `asombro | disgusto | enojo | feliz | normal | tristeza`

> **Instrucciones:** Ejecutar celdas en orden. Requiere GPU (Entorno de ejecución → T4 GPU)
"""

# CELDA 1 — Montar Google Drive
from google.colab import drive
drive.mount('/content/drive')
print('✅ Drive montado')

# CELDA 2 — Instalar dependencias + reinicio automático
# IMPORTANTE: Esta celda se ejecuta UNA SOLA VEZ.
# Después del reinicio automático, ejecuta desde Celda 3 en adelante.
```

```
import subprocess, sys

subprocess.run([
    sys.executable, '-m', 'pip', 'install',
    'mediapipe==0.10.21', '--quiet'
], check=True)

print('✅ Mediapipe instalado')
print('🔄 Reiniciando entorno para limpiar conflicto binario de numpy...')
print('   Colab se reconecta solo — espera unos segundos y ejecuta desde Celda 3')

import os
os.kill(os.getpid(), 9)  # Mata el proceso → Colab reinicia el kernel automáticamente

# CELDA 3 — Importaciones
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (classification_report, confusion_matrix,
                             accuracy_score, f1_score)

import tensorflow as tf
from tensorflow.keras.models import Model, Sequential, load_model
from tensorflow.keras.layers import (Dense, Dropout, Flatten, Conv2D, MaxPooling2D,
                                     BatchNormalization, GlobalAveragePooling2D, Input)
from tensorflow.keras.regularizers import l2
from tensorflow.keras.applications import MobileNetV2
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical

import mediapipe as mp

print(f'TensorFlow: {tf.__version__}')
print(f'GPU disponible: {len(tf.config.list_physical_devices("GPU")) > 0}')
print('✅ Importaciones completas')

# CELDA 4 — Configuración de rutas y constantes
RUTA_DATASET     = '/content/drive/MyDrive/base_de_datos/dataimages'
RUTA_DRIVE       = '/content/drive/MyDrive'

# Modelos CNN
RUTA_CNN_BASE    = os.path.join(RUTA_DRIVE, 'modelo_cnn_base.keras')
RUTA_CNN_MEJOR   = os.path.join(RUTA_DRIVE, 'modelo_cnn_mobilenet.keras')

# Modelos ML
RUTA_SVM         = os.path.join(RUTA_DRIVE, 'modelo_svm_final.pkl')
RUTA_RF          = os.path.join(RUTA_DRIVE, 'modelo_rf_final.pkl')
RUTA_ENCODER     = os.path.join(RUTA_DRIVE, 'encoder_final.pkl')
RUTA_SCALER      = os.path.join(RUTA_DRIVE, 'scaler_final.pkl')

# CSV MediaPipe
RUTA_CSV_MP      = os.path.join(RUTA_DRIVE, 'features_mediapipe.csv')
RUTA_CSV_MLP     = os.path.join(RUTA_DRIVE, 'puntos_facial_emociones.csv')

IMG_SIZE         = (96, 96)
N_CLASES         = 6
RANDOM_STATE     = 42

assert os.path.exists(RUTA_DATASET), f'❌ No se encontró: {RUTA_DATASET}'
clases = sorted(os.listdir(RUTA_DATASET))
print(f'✅ Dataset encontrado. Clases: {clases}')

# CELDA 5 — Funciones utilitarias (disponibles para todos los modelos)
```

```
def graficar_historial(historial, titulo='Curvas de aprendizaje'):
    """Grafica accuracy y loss de un historial de entrenamiento Keras."""
    acc  = historial.history['accuracy']
    vacc = historial.history['val_accuracy']
    loss = historial.history['loss']
    vloss= historial.history['val_loss']
    epocas = range(1, len(acc) + 1)

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(13, 4))

    ax1.plot(epocas, acc,  'b-o', ms=3, label='Entrenamiento')
    ax1.plot(epocas, vacc, 'r-o', ms=3, label='Validación')
    ax1.set_title('Precisión por época', fontsize=12)
    ax1.set_xlabel('Época'); ax1.set_ylabel('Precisión')
    ax1.legend(); ax1.grid(True, alpha=0.3)

    ax2.plot(epocas, loss,  'b-o', ms=3, label='Entrenamiento')
    ax2.plot(epocas, vloss, 'r-o', ms=3, label='Validación')
    ax2.set_title('Pérdida por época', fontsize=12)
    ax2.set_xlabel('Época'); ax2.set_ylabel('Pérdida')
    ax2.legend(); ax2.grid(True, alpha=0.3)

    fig.suptitle(titulo, fontsize=13, fontweight='bold')
    plt.tight_layout()
    plt.show()

def graficar_confusion(y_true, y_pred, nombres, titulo, cmap='Blues'):
    """Grafica una matriz de confusión con seaborn."""
    cm  = confusion_matrix(y_true, y_pred)
    acc = accuracy_score(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap=cmap,
                xticklabels=nombres, yticklabels=nombres)
    plt.title(f'{titulo}\nAccuracy: {acc:.2%}', fontsize=13)
    plt.xlabel('Predicción'); plt.ylabel('Real')
    plt.tight_layout()
    plt.show()
    return cm, acc
```

```
print('✅ Funciones utilitarias definidas')


# CELDA 6 — Análisis exploratorio (EDA)
conteo = {}
for clase in clases:
    carpeta = os.path.join(RUTA_DATASET, clase)
    if os.path.isdir(carpeta):
        archivos = [f for f in os.listdir(carpeta)
                    if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
        conteo[clase] = len(archivos)

total = sum(conteo.values())
print(f'Total imágenes: {total}')
for c, n in conteo.items():
    print(f'  {c:12s}: {n} imágenes')

colores = ['#E74C3C','#E67E22','#F1C40F','#2ECC71','#3498DB','#9B59B6']
plt.figure(figsize=(9, 4))
plt.bar(conteo.keys(), conteo.values(), color=colores, edgecolor='black')
plt.title('Distribución de imágenes por emoción', fontsize=14)
plt.xlabel('Emoción'); plt.ylabel('Cantidad')
plt.xticks(rotation=15)
for i, (k, v) in enumerate(conteo.items()):
    plt.text(i, v + 1, str(v), ha='center', fontsize=10, fontweight='bold')
plt.tight_layout()
plt.show()
print('✅ EDA completado')

# CELDA 7 — Muestra de imágenes por clase
fig, axes = plt.subplots(2, 3, figsize=(12, 8))
fig.suptitle('Ejemplo de imágenes por emoción', fontsize=15, fontweight='bold')
for ax, clase in zip(axes.flatten(), clases):
    carpeta = os.path.join(RUTA_DATASET, clase)
    archivos = [f for f in os.listdir(carpeta)
                if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
    if archivos:
        img = cv2.imread(os.path.join(carpeta, archivos[0]))
```

```
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (200, 200))
        ax.imshow(img)
        ax.set_title(clase.upper(), fontsize=12, fontweight='bold')
    ax.axis('off')
plt.tight_layout()
plt.show()

"""---
## Parte 1A — CNN Base (Arquitectura Original de la Tesis)

Arquitectura documentada en la tesis:
- **Detección facial:** Haar Cascade (OpenCV)
- **Entrada:** 96×96 píxeles en escala de grises
- **3 capas convolucionales:** 32 → 64 → 128 filtros
- **Regularización:** Dropout 0.5
- **Optimizador:** Adam | **Épocas:** hasta 25 con EarlyStopping
"""

# CELDA 8 — Cargar dataset para CNN Base (Haar Cascade + grises)
def cargar_dataset_cnn_base(ruta, tamaño=(96, 96)):
    """
    Carga imágenes, detecta rostros con Haar Cascade y devuelve arrays numpy.
    Reproducción exacta del enfoque documentado en la tesis.
    """
    face_cascade = cv2.CascadeClassifier(
        cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
    )
    X, y = [], []
    clases_locales = sorted([
        c for c in os.listdir(ruta) if os.path.isdir(os.path.join(ruta, c))
    ])
    sin_rostro = 0

    for clase in clases_locales:
        carpeta = os.path.join(ruta, clase)
        archivos = [f for f in os.listdir(carpeta)
                    if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
```

```
        detectados = 0
        for archivo in archivos:
            img = cv2.imread(os.path.join(carpeta, archivo))
            if img is None:
                continue
            gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            rostros = face_cascade.detectMultiScale(
                gris, scaleFactor=1.1, minNeighbors=5
            )
            if len(rostros) > 0:
                x_, y_, w, h = rostros[0]
                rostro = gris[y_:y_+h, x_:x_+w]
                rostro = cv2.resize(rostro, tamaño).astype('float32') / 255.0
                X.append(rostro)
                y.append(clase)
                detectados += 1
            else:
                sin_rostro += 1
        print(f'  {clase:12s}: {detectados}/{len(archivos)} detectados')

    X = np.expand_dims(np.array(X), axis=-1)  # (N, 96, 96, 1)
    print(f'\n  Sin rostro detectado: {sin_rostro}')
    return X, np.array(y)

print('⚙️  Cargando dataset CNN Base (Haar Cascade)...')
X_base, y_base_str = cargar_dataset_cnn_base(RUTA_DATASET)

# Codificar etiquetas
enc_base = LabelEncoder()
y_base_enc = enc_base.fit_transform(y_base_str)
y_base_cat = to_categorical(y_base_enc)
CLASES_BASE = enc_base.classes_

# Split estratificado 80/20
X_b_train, X_b_test, y_b_train, y_b_test = train_test_split(
    X_base, y_base_cat, test_size=0.20,
    random_state=RANDOM_STATE, stratify=y_base_enc
)
```

```
print(f'\nTotal imágenes: {len(X_base)}')
print(f'Train: {len(X_b_train)} | Test: {len(X_b_test)}')
print(f'Clases: {list(CLASES_BASE)}')
print('✅ Dataset CNN Base listo')

# CELDA 9 — Construir CNN Base (arquitectura documentada en la tesis)
def construir_cnn_base(n_clases, img_size=(96, 96)):
    """
    Tabla 1 de la tesis:
    Conv2D(32) → MaxPool → Conv2D(64) → MaxPool → Conv2D(128) → MaxPool
    → Flatten → Dense(128, ReLU) → Dropout(0.5) → Dense(n_clases, Softmax)
    """
    modelo = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(*img_size, 1)),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(n_clases, activation='softmax')
    ])
    modelo.compile(
        optimizer=Adam(learning_rate=1e-3),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return modelo

modelo_cnn_base = construir_cnn_base(N_CLASES)
modelo_cnn_base.summary()
print('✅ CNN Base construida')

# CELDA 10 — Entrenar CNN Base
callbacks_base = [
```

```
    EarlyStopping(monitor='val_accuracy', patience=8,
                  restore_best_weights=True, verbose=1),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5,
                      patience=4, min_lr=1e-6, verbose=1),
    ModelCheckpoint(RUTA_CNN_BASE, monitor='val_accuracy',
                    save_best_only=True, verbose=1)
]

print('🔵 Entrenando CNN Base...')
hist_base = modelo_cnn_base.fit(
    X_b_train, y_b_train,
    validation_data=(X_b_test, y_b_test),
    epochs=25,
    batch_size=16,
    callbacks=callbacks_base,
    verbose=1
)
best_base = max(hist_base.history['val_accuracy'])
print(f'\n✅ CNN Base completada — mejor val_accuracy: {best_base:.4f}')

# CELDA 11 — Evaluar CNN Base
modelo_cnn_base = load_model(RUTA_CNN_BASE)

y_pred_base = np.argmax(modelo_cnn_base.predict(X_b_test, verbose=0), axis=1)
y_true_base = np.argmax(y_b_test, axis=1)

acc_cnn_base = accuracy_score(y_true_base, y_pred_base)
f1_cnn_base  = f1_score(y_true_base, y_pred_base, average='weighted')

print(f'📊 CNN Base — Accuracy: {acc_cnn_base:.4f} | F1-score: {f1_cnn_base:.4f}')
print('\nReporte por clase:')
print(classification_report(y_true_base, y_pred_base, target_names=CLASES_BASE))

# Curvas de aprendizaje
graficar_historial(hist_base, titulo='CNN Base — Curvas de aprendizaje')

# Matriz de confusión
graficar_confusion(y_true_base, y_pred_base, CLASES_BASE,
```

```
                    'CNN Base (3 capas Conv2D)', cmap='Purples')
print('✅ Evaluación CNN Base completada')


"""---
## Parte 1B — CNN Mejorada (Transfer Learning con MobileNetV2)

Mejoras respecto a la CNN Base:
- **Base preentrenada:** MobileNetV2 (ImageNet ILSVRC-2012, ~1.28M imágenes, 53 capas)
- **Entrada:** 96×96 píxeles RGB (color)
- **Data augmentation:** rotación, zoom, flip horizontal, cambios de brillo
- **Entrenamiento en 2 fases:**
  - Fase 1: base congelada, solo se entrena la cabeza clasificadora
  - Fase 2: fine-tuning de las últimas 30 capas con lr muy bajo
"""


# CELDA 12 — Preparar dataset con Haar Cascade para MobileNetV2 (RGB)
  def cargar_dataset_cnn_rgb(ruta, tamaño=(96, 96)):
      """Haar Cascade + recorte → RGB 96×96 normalizado."""
      face_cascade = cv2.CascadeClassifier(
          cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
      )
      X, y = [], []
      clases_locales = sorted([
          c for c in os.listdir(ruta) if os.path.isdir(os.path.join(ruta, c))
      ])
      sin_rostro = 0
      for clase in clases_locales:
          carpeta = os.path.join(ruta, clase)
          archivos = [f for f in os.listdir(carpeta)
                      if f.lower().endswith(('.jpg', '.jpeg', '.png'))]
          detectados = 0
          for archivo in archivos:
              img = cv2.imread(os.path.join(carpeta, archivo))
              if img is None:
                  continue
              gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
              rostros = face_cascade.detectMultiScale(gris, 1.1, 5)
              if len(rostros) > 0:
```

```
                x_, y_, w, h = rostros[0]
                rostro = img[y_:y_+h, x_:x_+w]
                rostro = cv2.cvtColor(rostro, cv2.COLOR_BGR2RGB)
                rostro = cv2.resize(rostro, tamaño).astype('float32')
                rostro = tf.keras.applications.mobilenet_v2.preprocess_input(rostro)
                X.append(rostro)
                y.append(clase)
                detectados += 1
            else:
                sin_rostro += 1
        print(f'  {clase:12s}: {detectados}/{len(archivos)} detectados')

    print(f'\n  Sin rostro: {sin_rostro}')
    return np.array(X), np.array(y)

print('⚙️  Cargando dataset RGB con Haar Cascade...')
X_rgb, y_rgb_str = cargar_dataset_cnn_rgb(RUTA_DATASET)

enc_rgb = LabelEncoder()
y_rgb_enc = enc_rgb.fit_transform(y_rgb_str)
y_rgb_cat = to_categorical(y_rgb_enc)
CLASES_RGB = enc_rgb.classes_

X_r_train, X_r_test, y_r_train, y_r_test = train_test_split(
    X_rgb, y_rgb_cat, test_size=0.20,
    random_state=RANDOM_STATE, stratify=y_rgb_enc
)

datagen_rgb = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.10,
    height_shift_range=0.10,
    zoom_range=0.15,
    horizontal_flip=True,
    brightness_range=[0.75, 1.25],
    fill_mode='nearest'
)
datagen_rgb.fit(X_r_train)
```

```
  print(f'\nTotal: {len(X_rgb)} | Train: {len(X_r_train)} | Test: {len(X_r_test)}')
  print(f'Clases: {list(CLASES_RGB)}')
  print('✅ Dataset RGB listo')

# CELDA 13 — Construir modelo MobileNetV2
def construir_modelo_mobilenet(n_clases, img_size, lr=1e-3):
    base = MobileNetV2(
        input_shape=(*img_size, 3),
        include_top=False,
        weights='imagenet'
    )
    base.trainable = False

    entradas = Input(shape=(*img_size, 3))
    x = base(entradas, training=False)
    x = GlobalAveragePooling2D()(x)
    x = Dense(256, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.50)(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.30)(x)
    salidas = Dense(n_clases, activation='softmax')(x)

    modelo = Model(entradas, salidas)
    modelo.compile(
        optimizer=Adam(learning_rate=lr),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return modelo, base

modelo_cnn_mejor, base_mobilenet = construir_modelo_mobilenet(N_CLASES, IMG_SIZE)
modelo_cnn_mejor.summary()
print('✅ MobileNetV2 construido')

# CELDA 14 — Entrenamiento FASE 1 (cabeza clasificadora)
  callbacks_f1 = [
```

```
    EarlyStopping(monitor='val_accuracy', patience=8,
                  restore_best_weights=True, verbose=1),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5,
                      patience=4, min_lr=1e-6, verbose=1),
    ModelCheckpoint(RUTA_CNN_MEJOR, monitor='val_accuracy',
                    save_best_only=True, verbose=1)
]

print('🟠  Fase 1: Entrenando cabeza clasificadora (base congelada)...')
hist_f1 = modelo_cnn_mejor.fit(
    datagen_rgb.flow(X_r_train, y_r_train, batch_size=32),
    validation_data=(X_r_test, y_r_test),
    epochs=35, callbacks=callbacks_f1, verbose=1
)
print(f'\n✅ Fase 1 — mejor val_accuracy: {max(hist_f1.history["val_accuracy"]):.4f}')

# CELDA 15 — Entrenamiento FASE 2 (fine-tuning últimas 30 capas)
base_mobilenet.trainable = True
for capa in base_mobilenet.layers[:-30]:
    capa.trainable = False

modelo_cnn_mejor.compile(
    optimizer=Adam(learning_rate=2e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

callbacks_f2 = [
    EarlyStopping(monitor='val_accuracy', patience=10,
                  restore_best_weights=True, verbose=1),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5,
                      patience=5, min_lr=1e-7, verbose=1),
    ModelCheckpoint(RUTA_CNN_MEJOR, monitor='val_accuracy',
                    save_best_only=True, verbose=1)
]

print('🔴 Fase 2: Fine-tuning últimas 30 capas...')
hist_f2 = modelo_cnn_mejor.fit(
```

```
      datagen_rgb.flow(X_r_train, y_r_train, batch_size=32),
      validation_data=(X_r_test, y_r_test),
      epochs=25, callbacks=callbacks_f2, verbose=1
  )
  print(f'\n✅ Fase 2 — mejor val_accuracy: {max(hist_f2.history["val_accuracy"]):.4f}')

# CELDA 16 — Curvas de aprendizaje CNN Mejorada (Fase 1 + Fase 2 juntas)
acc  = hist_f1.history['accuracy']  + hist_f2.history['accuracy']
vacc = hist_f1.history['val_accuracy'] + hist_f2.history['val_accuracy']
loss = hist_f1.history['loss']  + hist_f2.history['loss']
vloss= hist_f1.history['val_loss'] + hist_f2.history['val_loss']
offset = len(hist_f1.history['accuracy'])
epocas = range(1, len(acc) + 1)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(13, 4))
ax1.plot(epocas, acc,  'b-o', ms=3, label='Entrenamiento')
ax1.plot(epocas, vacc, 'r-o', ms=3, label='Validación')
ax1.axvline(x=offset, color='gray', linestyle='--', label='Inicio fine-tuning')
ax1.set_title('Precisión por época'); ax1.set_xlabel('Época'); ax1.set_ylabel('Precisión')
ax1.legend(); ax1.grid(True, alpha=0.3)

ax2.plot(epocas, loss,  'b-o', ms=3, label='Entrenamiento')
ax2.plot(epocas, vloss, 'r-o', ms=3, label='Validación')
ax2.axvline(x=offset, color='gray', linestyle='--', label='Inicio fine-tuning')
ax2.set_title('Pérdida por época'); ax2.set_xlabel('Época'); ax2.set_ylabel('Pérdida')
ax2.legend(); ax2.grid(True, alpha=0.3)

fig.suptitle('CNN Mejorada (MobileNetV2) — Fase 1 + Fase 2', fontsize=13, fontweight='bold')
plt.tight_layout()
plt.show()

# CELDA 17 — Evaluar CNN Mejorada
  modelo_cnn_mejor = load_model(RUTA_CNN_MEJOR)

  y_pred_mejor = np.argmax(modelo_cnn_mejor.predict(X_r_test, verbose=0), axis=1)
  y_true_mejor = np.argmax(y_r_test, axis=1)

  acc_cnn_mejor = accuracy_score(y_true_mejor, y_pred_mejor)
```

```
  f1_cnn_mejor  = f1_score(y_true_mejor, y_pred_mejor, average='weighted')

  print(f'📊 CNN Mejorada — Accuracy: {acc_cnn_mejor:.4f} | F1-score: {f1_cnn_mejor:.4f}')
  print('\nReporte por clase:')
  print(classification_report(y_true_mejor, y_pred_mejor, target_names=CLASES_RGB))

  # Curvas Fase 1 + Fase 2
  acc  = hist_f1.history['accuracy']  + hist_f2.history['accuracy']
  vacc = hist_f1.history['val_accuracy'] + hist_f2.history['val_accuracy']
  loss = hist_f1.history['loss']  + hist_f2.history['loss']
  vloss= hist_f1.history['val_loss'] + hist_f2.history['val_loss']
  offset = len(hist_f1.history['accuracy'])
  epocas = range(1, len(acc) + 1)

  fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(13, 4))
  ax1.plot(epocas, acc,  'b-o', ms=3, label='Entrenamiento')
  ax1.plot(epocas, vacc, 'r-o', ms=3, label='Validación')
  ax1.axvline(x=offset, color='gray', linestyle='--', label='Inicio fine-tuning')
  ax1.set_title('Precisión por época'); ax1.set_xlabel('Época'); ax1.set_ylabel('Precisión')
  ax1.legend(); ax1.grid(True, alpha=0.3)

  ax2.plot(epocas, loss,  'b-o', ms=3, label='Entrenamiento')
  ax2.plot(epocas, vloss, 'r-o', ms=3, label='Validación')
  ax2.axvline(x=offset, color='gray', linestyle='--', label='Inicio fine-tuning')
  ax2.set_title('Pérdida por época'); ax2.set_xlabel('Época'); ax2.set_ylabel('Pérdida')
  ax2.legend(); ax2.grid(True, alpha=0.3)

  fig.suptitle('CNN Mejorada (MobileNetV2) — Fase 1 + Fase 2', fontsize=13, fontweight='bold')
  plt.tight_layout(); plt.show()

  graficar_confusion(y_true_mejor, y_pred_mejor, CLASES_RGB,
                     'CNN Mejorada (MobileNetV2)', cmap='Blues')
  print('✅ Evaluación CNN Mejorada completada')

"""---
## Parte 2 — SVM y Random Forest con MediaPipe FaceMesh

Se extraen **11 features geométricas normalizadas** (ratios respecto al ancho del rostro).
```

```
Al ser ratios y no distancias absolutas, son **invariantes al tamaño de imagen y distancia a la cámara**.
"""

# CELDA 18 — Extracción de features con MediaPipe
def extraer_features_mediapipe(ruta_imagen, face_mesh):
    img = cv2.imread(ruta_imagen)
    if img is None:
        return None
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    resultado = face_mesh.process(img_rgb)
    if not resultado.multi_face_landmarks:
        return None

    h, w = img.shape[:2]
    lm = resultado.multi_face_landmarks[0].landmark
    pts = np.array([[p.x * w, p.y * h] for p in lm])

    def dist(a, b):
        return np.linalg.norm(pts[a] - pts[b])

    ancho_rostro  = dist(234, 454) + 1e-6
    ancho_boca    = dist(61, 291)  / ancho_rostro
    alto_boca     = dist(13, 14)   / ancho_rostro
    ratio_sonrisa = ancho_boca / (alto_boca + 1e-6)
    aper_ojo_izq  = dist(159, 145) / ancho_rostro
    aper_ojo_der  = dist(386, 374) / ancho_rostro
    aper_media    = (aper_ojo_izq + aper_ojo_der) / 2
    ceja_ojo_izq  = dist(70, 159)  / ancho_rostro
    ceja_ojo_der  = dist(300, 386) / ancho_rostro
    dist_entrecejo= dist(107, 336) / ancho_rostro
    asimetria     = abs(pts[61][1] - pts[291][1]) / ancho_rostro
    nariz_boca    = dist(4, 13)    / ancho_rostro

    return [ancho_boca, alto_boca, ratio_sonrisa,
            aper_ojo_izq, aper_ojo_der, aper_media,
            ceja_ojo_izq, ceja_ojo_der, dist_entrecejo,
            asimetria, nariz_boca]
```

```
def generar_csv_features(ruta_dataset, ruta_salida):
    mp_fm = mp.solutions.face_mesh
    fm = mp_fm.FaceMesh(static_image_mode=True, max_num_faces=1,
                        refine_landmarks=True, min_detection_confidence=0.5)
    cols = ['label','ancho_boca','alto_boca','ratio_sonrisa',
            'aper_ojo_izq','aper_ojo_der','aper_media',
            'ceja_ojo_izq','ceja_ojo_der','dist_entrecejo',
            'asimetria','nariz_boca']
    filas, sin_det = [], 0
    for emocion in sorted(os.listdir(ruta_dataset)):
        carpeta = os.path.join(ruta_dataset, emocion)
        if not os.path.isdir(carpeta): continue
        for nombre in os.listdir(carpeta):
            if not nombre.lower().endswith(('.jpg','.jpeg','.png')): continue
            feats = extraer_features_mediapipe(os.path.join(carpeta, nombre), fm)
            if feats: filas.append([emocion] + feats)
            else: sin_det += 1
    fm.close()
    df = pd.DataFrame(filas, columns=cols)
    df.to_csv(ruta_salida, index=False)
    print(f'✅ CSV guardado: {ruta_salida} ({len(df)} filas, {sin_det} sin rostro)')
    return df

if os.path.exists(RUTA_CSV_MP):
    print('📂 Cargando CSV MediaPipe existente...')
    df_mp = pd.read_csv(RUTA_CSV_MP)
    print(f'   {len(df_mp)} filas | Clases: {df_mp["label"].unique()}')
else:
    print('⚙️  Generando features MediaPipe (~3 min)...')
    df_mp = generar_csv_features(RUTA_DATASET, RUTA_CSV_MP)
print('✅ Features MediaPipe listas')

# CELDA 19 — Preparar datos ML
X_mp  = df_mp.drop(columns=['label']).values.astype('float32')
y_mp  = df_mp['label'].values

encoder = LabelEncoder()
```

```
y_enc   = encoder.fit_transform(y_mp)
joblib.dump(encoder, RUTA_ENCODER)

scaler   = StandardScaler()
X_scaled = scaler.fit_transform(X_mp)
joblib.dump(scaler, RUTA_SCALER)

X_ml_train, X_ml_test, y_ml_train, y_ml_test = train_test_split(
    X_scaled, y_enc, test_size=0.20,
    random_state=RANDOM_STATE, stratify=y_enc
)
X_rf_train, X_rf_test, y_rf_train, y_rf_test = train_test_split(
    X_mp, y_enc, test_size=0.20,
    random_state=RANDOM_STATE, stratify=y_enc
)

print(f'Clases: {list(encoder.classes_)}')
print(f'Train: {len(X_ml_train)} | Test: {len(X_ml_test)}')
print('✅ Datos ML preparados')

# CELDA 20 — Entrenar y evaluar SVM
print('🔵 Entrenando SVM...')
modelo_svm = SVC(kernel='rbf', C=10, gamma='scale',
                 probability=True, random_state=RANDOM_STATE)
modelo_svm.fit(X_ml_train, y_ml_train)

y_pred_svm = modelo_svm.predict(X_ml_test)
acc_svm = accuracy_score(y_ml_test, y_pred_svm)
f1_svm  = f1_score(y_ml_test, y_pred_svm, average='weighted')

print(f'\n📊 SVM — Accuracy: {acc_svm:.4f} | F1-score: {f1_svm:.4f}')
print(classification_report(y_ml_test, y_pred_svm, target_names=encoder.classes_))

# Validación cruzada
cv_svm = cross_val_score(modelo_svm, X_scaled, y_enc,
                         cv=StratifiedKFold(5, shuffle=True, random_state=RANDOM_STATE))
print(f'CV 5-fold: {cv_svm.mean():.4f} ± {cv_svm.std():.4f}')
```

```
graficar_confusion(y_ml_test, y_pred_svm, encoder.classes_,
                   'SVM (RBF kernel)', cmap='Greens')
joblib.dump(modelo_svm, RUTA_SVM)
print('✅ SVM guardado')

# CELDA 21 — Entrenar y evaluar Random Forest
print('🟢 Entrenando Random Forest...')
modelo_rf = RandomForestClassifier(
    n_estimators=300, class_weight='balanced',
    random_state=RANDOM_STATE, n_jobs=-1
)
modelo_rf.fit(X_rf_train, y_rf_train)

y_pred_rf = modelo_rf.predict(X_rf_test)
acc_rf = accuracy_score(y_rf_test, y_pred_rf)
f1_rf  = f1_score(y_rf_test, y_pred_rf, average='weighted')

print(f'\n📊 Random Forest — Accuracy: {acc_rf:.4f} | F1-score: {f1_rf:.4f}')
print(classification_report(y_rf_test, y_pred_rf, target_names=encoder.classes_))

# Importancia de features
feat_names = ['ancho_boca','alto_boca','ratio_sonrisa','aper_ojo_izq',
              'aper_ojo_der','aper_media','ceja_ojo_izq','ceja_ojo_der',
              'dist_entrecejo','asimetria','nariz_boca']
importancias = modelo_rf.feature_importances_
idx = np.argsort(importancias)[::-1]
plt.figure(figsize=(10, 4))
plt.bar(range(len(importancias)), importancias[idx], color='steelblue')
plt.xticks(range(len(importancias)), [feat_names[i] for i in idx], rotation=30, ha='right')
plt.title('Random Forest — Importancia de features geométricas', fontsize=12)
plt.ylabel('Importancia'); plt.tight_layout(); plt.show()

graficar_confusion(y_rf_test, y_pred_rf, encoder.classes_,
                   'Random Forest', cmap='Oranges')
joblib.dump(modelo_rf, RUTA_RF)
print('✅ Random Forest guardado')

"""---
```

```
## Parte 2B — MLP (Perceptrón Multicapa) con landmarks MediaPipe

A diferencia de SVM/RF (11 features geométricas), el MLP usa las **1404 coordenadas
crudas** (468 landmarks × 3 coords x,y,z) extraídas con MediaPipe FaceMesh.
- **Entrada:** `puntos_facial_emociones.csv` (1404 features)
- **Normalización:** StandardScaler
- **Regularización:** L2(0.20) + Dropout fuerte (0.8/0.7/0.6)
- **Épocas:** 15 | **Batch:** 32
"""

# CELDA 22 — Cargar dataset MLP (landmarks crudos)
np.random.seed(5)
tf.random.set_seed(5)

if os.path.exists(RUTA_CSV_MLP):
    print('📂 Cargando CSV de landmarks...')
    df_mlp = pd.read_csv(RUTA_CSV_MLP)
else:
    from google.colab import files as files_mlp
    print('📤 Sube puntos_facial_emociones.csv')
    uploaded_mlp = files_mlp.upload()
    nombre_mlp = list(uploaded_mlp.keys())[0]
    df_mlp = pd.read_csv(nombre_mlp)

X_mlp = df_mlp.drop(columns=['label']).values.astype('float32')
y_mlp = df_mlp['label'].values

scaler_mlp = StandardScaler()
X_mlp = scaler_mlp.fit_transform(X_mlp)

encoder_mlp = LabelEncoder()
y_mlp_enc = encoder_mlp.fit_transform(y_mlp)
y_mlp_cat = to_categorical(y_mlp_enc)
CLASES_MLP = encoder_mlp.classes_

X_mlp_train, X_mlp_test, y_mlp_train, y_mlp_test = train_test_split(
    X_mlp, y_mlp_cat, test_size=0.20,
    random_state=RANDOM_STATE
```

```
)

print(f'Features: {X_mlp.shape[1]} | Train: {len(X_mlp_train)} | Test: {len(X_mlp_test)}')
print(f'Clases: {list(CLASES_MLP)}')
print('✅ Dataset MLP listo')

# CELDA 23 — Construir y entrenar MLP
modelo_mlp = Sequential([
    Dense(512, activation='relu', input_shape=(X_mlp.shape[1],), kernel_regularizer=l2(0.20)),
    Dropout(0.8),
    Dense(256, activation='relu', kernel_regularizer=l2(0.20)),
    Dropout(0.7),
    Dense(128, activation='relu', kernel_regularizer=l2(0.20)),
    Dropout(0.6),
    Dense(y_mlp_cat.shape[1], activation='softmax')
])

modelo_mlp.compile(
    optimizer=Adam(learning_rate=1e-3),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

modelo_mlp.summary()

print('🟣 Entrenando MLP...')
hist_mlp = modelo_mlp.fit(
    X_mlp_train, y_mlp_train,
    validation_data=(X_mlp_test, y_mlp_test),
    epochs=15, batch_size=32, verbose=1
)
print(f'\n✅ MLP completado — val_accuracy: {hist_mlp.history["val_accuracy"][-1]:.4f}')

# CELDA 24 — Evaluar MLP
y_pred_mlp = np.argmax(modelo_mlp.predict(X_mlp_test, verbose=0), axis=1)
y_true_mlp = np.argmax(y_mlp_test, axis=1)

acc_mlp = accuracy_score(y_true_mlp, y_pred_mlp)
```

```
f1_mlp  = f1_score(y_true_mlp, y_pred_mlp, average='weighted')

print(f'📊 MLP — Accuracy: {acc_mlp:.4f} | F1-score: {f1_mlp:.4f}')
print('\nReporte por clase:')
print(classification_report(y_true_mlp, y_pred_mlp, target_names=CLASES_MLP))

# Curvas de aprendizaje
graficar_historial(hist_mlp, titulo='MLP (Perceptrón Multicapa) — Curvas de aprendizaje')

# Matriz de confusión
graficar_confusion(y_true_mlp, y_pred_mlp, CLASES_MLP,
                   'MLP (Perceptrón Multicapa)', cmap='YlOrRd')
print('✅ Evaluación MLP completada')

"""---
## Parte 3 — Comparación de los 5 Modelos
"""

# CELDA 25 — Tabla comparativa de los 5 modelos
resultados = pd.DataFrame({
    'Modelo': [
        'CNN Base (3 capas Conv2D)',
        'CNN Mejorada (MobileNetV2)',
        'SVM (RBF kernel)',
        'Random Forest',
        'MLP (Perceptrón Multicapa)'
    ],
    'Entrada': [
        'Imagen 96×96 grises',
        'Imagen 96×96 RGB + augmentación',
        'Features geométricas (11)',
        'Features geométricas (11)',
        'Landmarks MediaPipe (1404)'
    ],
    'Accuracy':             [acc_cnn_base, acc_cnn_mejor, acc_svm, acc_rf, acc_mlp],
    'F1-score (weighted)':  [f1_cnn_base,  f1_cnn_mejor,  f1_svm, f1_rf, f1_mlp]
})
```

```
print('\n' + '='*65)
print('       COMPARACIÓN FINAL DE MODELOS')
print('='*65)
df_print = resultados.copy()
df_print['Accuracy']           = df_print['Accuracy'].map('{:.2%}'.format)
df_print['F1-score (weighted)']= df_print['F1-score (weighted)'].map('{:.4f}'.format)
print(df_print.to_string(index=False))
print('='*65)

# Gráfico comparativo
modelos_nombres = ['CNN Base', 'CNN MobileNetV2', 'SVM', 'Random Forest', 'MLP']
acc_vals = [acc_cnn_base, acc_cnn_mejor, acc_svm, acc_rf, acc_mlp]
f1_vals  = [f1_cnn_base,  f1_cnn_mejor,  f1_svm, f1_rf, f1_mlp]
colores_m = ['#9B59B6', '#3498DB', '#2ECC71', '#E67E22', '#E74C3C']

x = np.arange(len(modelos_nombres)); w = 0.35
fig, ax = plt.subplots(figsize=(13, 5))
bars1 = ax.bar(x - w/2, acc_vals, w, label='Accuracy',    color=colores_m, edgecolor='black',
alpha=0.85)
bars2 = ax.bar(x + w/2, f1_vals,    w, label='F1-score',    color=colores_m, edgecolor='black',
alpha=0.50)

for bar in bars1:
    ax.text(bar.get_x()+bar.get_width()/2, bar.get_height()+0.005,
            f'{bar.get_height():.2%}', ha='center', va='bottom', fontsize=9)
for bar in bars2:
    ax.text(bar.get_x()+bar.get_width()/2, bar.get_height()+0.005,
            f'{bar.get_height():.3f}', ha='center', va='bottom', fontsize=9)

ax.set_ylim(0, 1.18)
ax.set_title('Comparación de modelos — Accuracy y F1-score', fontsize=13)
ax.set_xticks(x); ax.set_xticklabels(modelos_nombres, fontsize=11)
ax.legend(fontsize=11); ax.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.show()
print('✅ Comparación completada')

"""---
```

```
## Parte 4 — Demo de Predicción

Sube una imagen y los modelos la clasifican. Se muestran resultados de los 4 modelos CNN/SVM/RF.
"""

# CELDA 26 — Función de predicción (CNN Base, MobileNetV2, SVM, RF)
from google.colab import files

# Cargar modelos guardados
modelo_cnn_base  = load_model(RUTA_CNN_BASE)
modelo_cnn_mejor = load_model(RUTA_CNN_MEJOR)
modelo_svm       = joblib.load(RUTA_SVM)
modelo_rf        = joblib.load(RUTA_RF)
encoder          = joblib.load(RUTA_ENCODER)
scaler           = joblib.load(RUTA_SCALER)

face_cascade_pred = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
)
mp_fm_pred = mp.solutions.face_mesh.FaceMesh(
    static_image_mode=True, max_num_faces=1,
    refine_landmarks=True, min_detection_confidence=0.5
)

def predecir_todos(ruta_imagen):
    """Predice la emoción con los 4 modelos y muestra resultados comparativos."""
    img_orig = cv2.imread(ruta_imagen)
    if img_orig is None:
        print('❌ No se pudo leer la imagen'); return

    img_rgb  = cv2.cvtColor(img_orig, cv2.COLOR_BGR2RGB)
    img_draw = img_orig.copy()
    resultados_pred = {}

    # ---- CNN Base (grises, Haar Cascade) ----
    gris = cv2.cvtColor(img_orig, cv2.COLOR_BGR2GRAY)
    rostros = face_cascade_pred.detectMultiScale(gris, 1.1, 5)
    if len(rostros) > 0:
```

```
    x, y, w, h = rostros[0]
    crop_g = gris[y:y+h, x:x+w]
    crop_g = cv2.resize(crop_g, IMG_SIZE).astype('float32') / 255.0
    crop_g = np.expand_dims(crop_g, axis=(0, -1))
    proba_base = modelo_cnn_base.predict(crop_g, verbose=0)[0]
    resultados_pred['CNN Base'] = (enc_base.classes_, proba_base)
    cv2.rectangle(img_draw, (x, y), (x+w, y+h), (150, 0, 200), 3)

# ---- CNN Mejorada (RGB, MobileNetV2) ----
if len(rostros) > 0:
    crop_c = img_rgb[y:y+h, x:x+w]
    crop_c = cv2.resize(crop_c, IMG_SIZE).astype('float32') / 255.0
    crop_c = np.expand_dims(crop_c, axis=0)
    proba_mejor = modelo_cnn_mejor.predict(crop_c, verbose=0)[0]
    resultados_pred['CNN MobileNetV2'] = (list(CLASES_RGB.values()), proba_mejor)

# ---- SVM y Random Forest (MediaPipe) ----
feats = extraer_features_mediapipe(ruta_imagen, mp_fm_pred)
if feats is not None:
    vec = np.array(feats).reshape(1, -1)
    vec_sc = scaler.transform(vec)
    proba_svm = modelo_svm.predict_proba(vec_sc)[0]
    proba_rf  = modelo_rf.predict_proba(vec)[0]
    resultados_pred['SVM'] = (list(encoder.classes_), proba_svm)
    resultados_pred['Random Forest'] = (list(encoder.classes_), proba_rf)

if not resultados_pred:
    print('❌ No se detectó rostro en la imagen'); return

# ---- Visualización ----
n_modelos = len(resultados_pred)
fig, axes = plt.subplots(1, n_modelos + 1, figsize=(5 * (n_modelos + 1), 5))

axes[0].imshow(cv2.cvtColor(img_draw, cv2.COLOR_BGR2RGB))
axes[0].set_title('Imagen analizada', fontsize=12)
axes[0].axis('off')

colores_bar = ['#9B59B6', '#3498DB', '#2ECC71', '#E67E22']
```

```
    for ax, (nombre, (clases_m, proba)), color in zip(
            axes[1:], resultados_pred.items(), colores_bar):
        idx_pred = np.argmax(proba)
        ax.barh(range(len(clases_m)), proba, color=color, alpha=0.8)
        ax.set_yticks(range(len(clases_m)))
        ax.set_yticklabels(clases_m, fontsize=10)
        ax.set_xlim(0, 1)
        ax.set_title(f'{nombre}\n→ {clases_m[idx_pred].upper()} ({proba[idx_pred]:.1%})',
                     fontsize=10, fontweight='bold')
        ax.grid(axis='x', alpha=0.3)

    plt.suptitle('Predicción de emoción — 4 modelos', fontsize=13, fontweight='bold')
    plt.tight_layout()
    plt.show()

print('✅ Función de predicción lista')

# CELDA 27 — Subir imagen y predecir
print('📤 Selecciona una imagen...')
subida = files.upload()
for nombre_archivo in subida.keys():
    print(f'\n🔍 Analizando: {nombre_archivo}')
    predecir_todos(nombre_archivo)

# CELDA 28 — (Opcional) Probar con imagen del dataset
IMGEN_TEST = '/content/drive/MyDrive/base_de_datos/dataimages/feliz/Feliz_01_Maya.jpg'
if os.path.exists(IMGEN_TEST):
    print(f'🔍 Analizando: {os.path.basename(IMGEN_TEST)}')
    predecir_todos(IMGEN_TEST)
else:
    print(f'⚠️  Cambia IMGEN_TEST a una ruta válida de tu dataset.')
```

Anexo B — Código del detector en tiempo real

B.1 — Servidor (app.py)

```
"""
Detector de Emociones en Tiempo Real — 4 Modelos
UNAM — Facultad de Ingeniería — Ingeniería Mecatrónica — 2026
Guerrero Martínez Erick & Salgado Miranda Maya Jily
"""

import os
import copy
import cv2
import numpy as np
import pickle
import threading
import time
import warnings
from collections import deque
from datetime import datetime
from flask import Flask, Response, jsonify, render_template

warnings.filterwarnings("ignore")
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
import tensorflow as tf

import mediapipe as mp
from mediapipe.tasks.python import vision, BaseOptions
from mediapipe.tasks.python.vision import FaceLandmarker, FaceLandmarkerOptions, RunningMode

# ------------------------------------------------------------------------
MODEL_DIR = os.path.join(os.path.dirname(__file__), "models")
EMOCIONES = ["asombro", "disgusto", "enojo", "feliz", "normal", "tristeza"]
IMG_SIZE  = 96
CONF_UMBRAL = 0.70
CAMARA_INDEX = 0
CAM_WIDTH, CAM_HEIGHT, CAM_FPS = 1280, 720, 60

# ------------------------------------------------------------------------
# Cargar modelos
# ------------------------------------------------------------------------
print("[INFO] Cargando modelos...")
cnn_base = cnn_mejor = modelo_svm = modelo_rf = scaler = None

try:
    p = os.path.join(MODEL_DIR, "modelo_cnn_base.keras")
    if os.path.exists(p):
        cnn_base = tf.keras.models.load_model(p)
        cnn_base(np.zeros((1,IMG_SIZE,IMG_SIZE,1), dtype=np.float32), training=False)
        print("[OK] CNN Base")
    p = os.path.join(MODEL_DIR, "modelo_cnn_mobilenet.keras")
    if os.path.exists(p):
        cnn_mejor = tf.keras.models.load_model(p)
        cnn_mejor(np.zeros((1,IMG_SIZE,IMG_SIZE,3), dtype=np.float32), training=False)
        print("[OK] MobileNetV2")
    for name, attr in [("modelo_svm_final.pkl","svm"), ("modelo_rf_final.pkl","rf"),
("scaler_final.pkl","scaler")]:
        fp = os.path.join(MODEL_DIR, name)
        if os.path.exists(fp):
            with open(fp,"rb") as f: obj = pickle.load(f)
            if attr == "svm": modelo_svm = obj
            elif attr == "rf": modelo_rf = obj
            else: scaler = obj
            print(f"[OK] {name}")
except Exception as e:
    print(f"[ERROR] {e}")
```

```
cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

# Landmarks indices
LM = {"boca_izq":61,"boca_der":291,"boca_sup":13,"boca_inf":14,
      "ojo_izq_sup":159,"ojo_izq_inf":145,"ojo_der_sup":386,"ojo_der_inf":374,
      "ceja_izq":70,"ceja_der":300,"entrecejo_izq":107,"entrecejo_der":336,
      "nariz":4,"rostro_izq":234,"rostro_der":454}

def extraer_features(lms, w, h):
    pts = np.array([[l.x*w, l.y*h] for l in lms])
    d = lambda a,b: np.linalg.norm(pts[a]-pts[b])
    ar = d(LM["rostro_izq"],LM["rostro_der"])+1e-6
    ab = d(LM["boca_izq"],LM["boca_der"])/ar
    alb = d(LM["boca_sup"],LM["boca_inf"])/ar
    return np.array([ab,alb,ab/(alb+1e-6),
        d(LM["ojo_izq_sup"],LM["ojo_izq_inf"])/ar, d(LM["ojo_der_sup"],LM["ojo_der_inf"])/ar,
        (d(LM["ojo_izq_sup"],LM["ojo_izq_inf"])+d(LM["ojo_der_sup"],LM["ojo_der_inf"]))/(2*ar),
        d(LM["ceja_izq"],LM["ojo_izq_sup"])/ar, d(LM["ceja_der"],LM["ojo_der_sup"])/ar,
        d(LM["entrecejo_izq"],LM["entrecejo_der"])/ar,
        abs(pts[LM["boca_izq"]][1]-pts[LM["boca_der"]][1])/ar,
        d(LM["nariz"],LM["boca_sup"])/ar], dtype=np.float32)

# ---------------------------------------------------------------------------
# Estado global
# ---------------------------------------------------------------------------
lock = threading.Lock()
state = {k: {"emocion":"—","confianza":0.0,"probs":{e:0.0 for e in EMOCIONES}}
         for k in ["cnn_base","cnn_mejor","svm","rf"]}
state["fps"] = 0.0
state["rostro_detectado"] = False

# --- Logs de concordancia ---
LOG_MAX = 500
prediction_log = deque(maxlen=LOG_MAX)
agreement_stats = {"total": 0, "all_agree": 0, "majority": {e: 0 for e in EMOCIONES},
                   "per_model": {m: {"correct_majority": 0, "total": 0} for m in
["cnn_base","cnn_mejor","svm","rf"]}}

stream_lock = threading.Lock()
SMOOTH_N = 8
smoothers = {k: deque(maxlen=SMOOTH_N) for k in ["cnn_base","cnn_mejor","svm","rf"]}

def suavizar(buf, probs):
    buf.append(probs)
    avg = {e: sum(p.get(e,0) for p in buf)/len(buf) for e in EMOCIONES}
    best = max(avg, key=avg.get)
    return best, avg[best], avg

def predecir_cnn(modelo, entrada):
    if modelo is None or entrada is None: return None
    pr = modelo(tf.constant(entrada), training=False)[0].numpy()
    i = int(np.argmax(pr))
    return EMOCIONES[i], float(pr[i]), {EMOCIONES[j]:float(pr[j]) for j in range(len(EMOCIONES))}

def predecir_ml(modelo, feats, use_scaler=False):
    if modelo is None or feats is None: return None
    v = feats.reshape(1,-1)
    if use_scaler and scaler: v = scaler.transform(v)
    pr = modelo.predict_proba(v)[0]
    i = int(np.argmax(pr))
    return EMOCIONES[i], float(pr[i]), {EMOCIONES[j]:float(pr[j]) for j in range(len(EMOCIONES))}

# ---------------------------------------------------------------------------
# FaceMesh contours
# ---------------------------------------------------------------------------
CONTOURS = set()
try:
    cn = vision.FaceLandmarksConnections
    for g in [cn.FACE_LANDMARKS_FACE_OVAL, cn.FACE_LANDMARKS_LEFT_EYE,
```

```
              cn.FACE_LANDMARKS_RIGHT_EYE, cn.FACE_LANDMARKS_LEFT_EYEBROW,
              cn.FACE_LANDMARKS_RIGHT_EYEBROW, cn.FACE_LANDMARKS_LIPS]:
        for c in g: CONTOURS.add((c.start, c.end))
    print(f"[OK] FaceMesh {len(CONTOURS)} contours")
except: pass

def dibujar_lm(frame, face_lms):
    if not face_lms: return
    h,w = frame.shape[:2]
    for fl in face_lms:
        pts = [(int(l.x*w),int(l.y*h)) for l in fl]
        for px,py in pts: cv2.circle(frame,(px,py),1,(0,200,150),-1)
        for a,b in CONTOURS:
            if a<len(pts) and b<len(pts): cv2.line(frame,pts[a],pts[b],(0,130,100),1)

def dibujar_bbox(frame, bbox, em, cf):
    if not bbox: return
    x,y,w,h = bbox
    col = (0,220,120) if cf >= CONF_UMBRAL else (60,60,220)
    cv2.rectangle(frame,(x,y),(x+w,y+h),col,2)
    ly = max(y,28)
    cv2.rectangle(frame,(x,ly-28),(x+w,ly),col,-1)
    cv2.putText(frame,f"{em} {cf*100:.0f}%",(x+4,ly-8),cv2.FONT_HERSHEY_SIMPLEX,0.65,(10,10,10),2)

# ------------------------------------------------------------------------
# Generador MJPEG
# ------------------------------------------------------------------------
def generar_frames():
    if not stream_lock.acquire(blocking=False):
        yield b"--frame\r\nContent-Type: text/plain\r\n\r\nStream en uso.\r\n"
        return
    cap = landmarker = None
    try:
        cap = cv2.VideoCapture(CAMARA_INDEX, cv2.CAP_DSHOW)
        cap.set(cv2.CAP_PROP_FRAME_WIDTH, CAM_WIDTH)
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, CAM_HEIGHT)
        cap.set(cv2.CAP_PROP_FPS, CAM_FPS)
        cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
        if not cap.isOpened():
            print("[ERROR] No se pudo abrir camara"); return
        print(f"[OK] Camara {int(cap.get(3))}x{int(cap.get(4))}")

        fm_path = os.path.join(MODEL_DIR, "face_landmarker.task")
        if os.path.exists(fm_path):
            landmarker = FaceLandmarker.create_from_options(FaceLandmarkerOptions(
                base_options=BaseOptions(model_asset_path=fm_path),
                running_mode=RunningMode.VIDEO, num_faces=1,
                min_face_detection_confidence=0.5, min_tracking_confidence=0.5))
            print("[OK] FaceLandmarker")

        fc, t0, bbox, ts = 0, time.perf_counter(), None, 0

        while True:
            ok, frame = cap.read()
            if not ok: time.sleep(0.01); continue
            fc += 1; ts += 33

            # MediaPipe landmarks
            face_lms_data = None
            if landmarker:
                try:
                    r = landmarker.detect_for_video(
                        mp.Image(image_format=mp.ImageFormat.SRGB,
                                 data=cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)), ts)
                    dibujar_lm(frame, r.face_landmarks)
                    if r.face_landmarks: face_lms_data = r.face_landmarks[0]
                except: pass

            # Predicciones cada 3 frames
```

```
            if fc % 3 == 0:
                hi,wi = frame.shape[:2]
                gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                caras = cascade.detectMultiScale(gris, 1.1, 6, minSize=(80,80))
                bbox = None

                res = {"cnn_base":None,"cnn_mejor":None,"svm":None,"rf":None}

                if len(caras)>0:
                    x,y,w,h = caras[0]
                    x1,y1,x2,y2 = max(0,x),max(0,y),min(wi,x+w),min(hi,y+h)
                    bbox = (x1,y1,x2-x1,y2-y1)
                    # CNN Base
                    rg = cv2.resize(gris[y1:y2,x1:x2],(IMG_SIZE,IMG_SIZE)).astype("float32")/255.0
                    res["cnn_base"] = predecir_cnn(cnn_base, rg[np.newaxis,...,np.newaxis])
                    # MobileNetV2
                    rc = cv2.cvtColor(frame[y1:y2,x1:x2], cv2.COLOR_BGR2RGB)
                    rc = cv2.resize(rc,(IMG_SIZE,IMG_SIZE)).astype("float32")
                    rc = tf.keras.applications.mobilenet_v2.preprocess_input(rc)
                    res["cnn_mejor"] = predecir_cnn(cnn_mejor, rc[np.newaxis])

                # SVM/RF con MediaPipe
                if face_lms_data:
                    feats = extraer_features(face_lms_data, wi, hi)
                    res["svm"] = predecir_ml(modelo_svm, feats, True)
                    res["rf"]  = predecir_ml(modelo_rf, feats, False)

                with lock:
                    state["rostro_detectado"] = bbox is not None or face_lms_data is not None
                    for k in ["cnn_base","cnn_mejor","svm","rf"]:
                        if res[k]:
                            em,cf,pb = res[k]
                            s_em,s_cf,s_pb = suavizar(smoothers[k], pb)
                            state[k] = {"emocion":s_em,"confianza":s_cf,"probs":s_pb}
                        elif (k in ["cnn_base","cnn_mejor"] and bbox is None) or \
                             (k in ["svm","rf"] and face_lms_data is None):
                            smoothers[k].clear()
                            state[k] = {"emocion":"—","confianza":0.0,"probs":{e:0.0 for e in
EMOCIONES}}

                    # --- Logging ---
                    preds = {k: state[k]["emocion"] for k in ["cnn_base","cnn_mejor","svm","rf"]
                             if state[k]["emocion"] != "—"}
                    if len(preds) >= 2:
                        emociones_pred = list(preds.values())
                        from collections import Counter
                        conteo = Counter(emociones_pred)
                        mayoria, _ = conteo.most_common(1)[0]
                        todos_igual = len(set(emociones_pred)) == 1
                        agreement_stats["total"] += 1
                        if todos_igual:
                            agreement_stats["all_agree"] += 1
                        agreement_stats["majority"][mayoria] =
agreement_stats["majority"].get(mayoria, 0) + 1
                        for m, em in preds.items():
                            agreement_stats["per_model"][m]["total"] += 1
                            if em == mayoria:
                                agreement_stats["per_model"][m]["correct_majority"] += 1
                        entry = {"time": datetime.now().strftime("%H:%M:%S"),
                                 "preds": dict(preds),
                                 "majority": mayoria,
                                 "all_agree": todos_igual}
                        prediction_log.append(entry)

            # Overlay bbox con RF (mejor modelo)
            with lock:
                if modelo_rf: em_s,cf_s = state["rf"]["emocion"],state["rf"]["confianza"]
                elif cnn_base: em_s,cf_s = state["cnn_base"]["emocion"],state["cnn_base"]["confianza"]
                else: em_s,cf_s = "—",0.0
```

```
            dibujar_bbox(frame, bbox, em_s, cf_s)

            # FPS
            el = time.perf_counter()-t0
            if el>0:
                with lock: state["fps"] = round(fc/el,1)
                if fc>=60: fc=0; t0=time.perf_counter()

            cv2.putText(frame,"UNAM - Reconocimiento de Emociones 2026",
                        (10,frame.shape[0]-12),cv2.FONT_HERSHEY_SIMPLEX,0.5,(180,180,180),1)
            _,buf = cv2.imencode(".jpg",frame,[cv2.IMWRITE_JPEG_QUALITY,80])
            yield b"--frame\r\nContent-Type: image/jpeg\r\n\r\n"+buf.tobytes()+b"\r\n"

    except GeneratorExit: pass
    except Exception as e: print(f"[ERROR] {e}")
    finally:
        if landmarker: landmarker.close()
        if cap: cap.release(); print("[INFO] Camara liberada")
        stream_lock.release()

# ------------------------------------------------------------------------
# Flask
# ------------------------------------------------------------------------
app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html", emociones=EMOCIONES)

@app.route("/video_feed")
def video_feed():
    return Response(generar_frames(), mimetype="multipart/x-mixed-replace; boundary=frame")

@app.route("/predictions")
def predictions():
    with lock:
        return jsonify({
            "cnn_base": copy.deepcopy(state["cnn_base"]),
            "cnn_mejor": copy.deepcopy(state["cnn_mejor"]),
            "svm": copy.deepcopy(state["svm"]),
            "rf": copy.deepcopy(state["rf"]),
            "fps": state["fps"],
            "rostro": state["rostro_detectado"],
            "modelos": {"base_ok":cnn_base is not None,"mejor_ok":cnn_mejor is not None,
                        "svm_ok":modelo_svm is not None,"rf_ok":modelo_rf is not None}
        })

@app.route("/logs")
def logs():
    with lock:
        total = agreement_stats["total"]
        if total == 0:
            return jsonify({"msg": "Sin datos aun, muestra tu rostro a la camara"})
        pct_agree = agreement_stats["all_agree"] / total * 100
        model_stats = {}
        for m, s in agreement_stats["per_model"].items():
            if s["total"] > 0:
                model_stats[m] = {
                    "aciertos_con_mayoria": s["correct_majority"],
                    "total": s["total"],
                    "pct": round(s["correct_majority"] / s["total"] * 100, 1)
                }
        return jsonify({
            "total_predicciones": total,
            "todos_de_acuerdo": agreement_stats["all_agree"],
            "pct_acuerdo_total": round(pct_agree, 1),
            "emocion_mas_detectada": max(agreement_stats["majority"],
key=agreement_stats["majority"].get)
```

```
                                    if any(v > 0 for v in agreement_stats["majority"].values()) else
"—",
            "concordancia_por_modelo": model_stats,
            "ultimas_10": list(prediction_log)[-10:]
        })

if __name__ == "__main__":
    print(f"\n{'='*55}")
    print("  Detector de Emociones — UNAM 2026 — 4 modelos")
    print(f"{'='*55}")
    print(f"  CNN Base:      {'OK' if cnn_base else 'no'}")
    print(f"  MobileNetV2:   {'OK' if cnn_mejor else 'no'}")
    print(f"  SVM:           {'OK' if modelo_svm else 'no'}")
    print(f"  Random Forest: {'OK' if modelo_rf else 'no'}")
    print(f"{'='*55}")
    print("  http://localhost:5000")
    print(f"{'='*55}\n")
    app.run(host="0.0.0.0", port=5000, debug=False, threaded=True)
```

B.2 — Interfaz web (index.html)

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Detector de Emociones — UNAM 2026</title>
  <style>
    *, *::before, *::after { box-sizing: border-box; margin: 0; padding: 0; }
    :root {
      --bg: #0d0f14; --surface: #151820; --surface2: #1c2030;
      --border: #252a38; --accent: #00e5a0; --accent2: #00b8ff;
      --accent3: #ff9f43; --accent4: #a855f7;
      --warn: #ff4d6a; --text: #e8ecf4; --text-muted: #6b7494;
    }
    html, body {
      height: 100%; background: var(--bg); color: var(--text);
      font-family: 'Segoe UI', system-ui, sans-serif; font-size: 13px; overflow: hidden;
    }
    .shell { display: grid; grid-template-rows: 48px 1fr 28px; height: 100vh; }
    header {
      display: flex; align-items: center; justify-content: space-between;
      padding: 0 16px; background: var(--surface); border-bottom: 1px solid var(--border);
    }
    .logo { font-weight: 700; font-size: 14px; }
    .logo span { color: var(--accent); }
    .badge {
      background: var(--surface2); border: 1px solid var(--border);
      border-radius: 16px; padding: 2px 10px; font-size: 10px; color: var(--text-muted);
    }
    .status-dot {
      width: 7px; height: 7px; border-radius: 50%; background: var(--warn);
      display: inline-block; margin-right: 5px; transition: background .4s;
    }
    .status-dot.live { background: var(--accent); box-shadow: 0 0 6px var(--accent); }
    main { display: grid; grid-template-columns: 1fr 380px; overflow: hidden; }
    .video-panel {
      display: flex; align-items: center; justify-content: center;
      background: #000; position: relative; overflow: hidden;
    }
    .video-panel img { max-width: 100%; max-height: 100%; object-fit: contain; }
    .fps-badge {
      position: absolute; top: 10px; left: 10px;
      background: rgba(0,0,0,.7); border: 1px solid var(--border);
      border-radius: 5px; padding: 2px 8px; font-size: 11px; color: var(--accent);
    }
    .no-face-badge {
      position: absolute; bottom: 16px; left: 50%; transform: translateX(-50%);
      background: rgba(255,77,106,.15); border: 1px solid var(--warn);
      border-radius: 5px; padding: 3px 12px; font-size: 11px; color: var(--warn); display: none;
    }
    .side-panel {
      background: var(--surface); border-left: 1px solid var(--border);
      display: flex; flex-direction: column; overflow-y: auto;
    }
    .section-header {
      padding: 8px 14px; font-size: 9px; font-weight: 700; letter-spacing: 1.5px;
      text-transform: uppercase; color: var(--text-muted);
      background: var(--surface2); border-bottom: 1px solid var(--border);
    }
    .model-card { padding: 10px 14px; border-bottom: 1px solid var(--border); }
    .model-title {
      font-size: 10px; font-weight: 700; letter-spacing: .8px; text-transform: uppercase;
      color: var(--text-muted); margin-bottom: 4px;
      display: flex; align-items: center; justify-content: space-between;
    }
    .model-tag { font-size: 9px; border-radius: 3px; padding: 1px 6px; font-weight: 600; }
```

```
    .tag-base  { background: rgba(0,229,160,.12); color: var(--accent); border: 1px solid
rgba(0,229,160,.3); }
    .tag-mejor { background: rgba(0,184,255,.12); color: var(--accent2); border: 1px solid
rgba(0,184,255,.3); }
    .tag-svm   { background: rgba(255,159,67,.12); color: var(--accent3); border: 1px solid
rgba(255,159,67,.3); }
    .tag-rf    { background: rgba(168,85,247,.12); color: var(--accent4); border: 1px solid
rgba(168,85,247,.3); }
    .model-desc { font-size: 10px; color: var(--text-muted); margin-bottom: 6px; }
    .model-desc strong { color: var(--text); }
    .emocion-display { display: flex; align-items: center; justify-content: space-between;
margin-bottom: 6px; }
    .emocion-nombre { font-size: 18px; font-weight: 800; text-transform: capitalize; }
    .emocion-conf { font-size: 20px; font-weight: 900; font-variant-numeric: tabular-nums; }
    .conf-bajo { color: var(--warn); }
    .conf-alto { color: var(--accent); }
    .conf-alto-azul { color: var(--accent2); }
    .conf-alto-naranja { color: var(--accent3); }
    .conf-alto-morado { color: var(--accent4); }
    .barras { display: flex; flex-direction: column; gap: 3px; }
    .barra-row { display: grid; grid-template-columns: 62px 1fr 32px; align-items: center; gap: 4px; }
    .barra-label { font-size: 10px; color: var(--text-muted); text-transform: capitalize; }
    .barra-track { height: 8px; background: var(--surface2); border-radius: 4px; overflow: hidden; }
    .barra-fill { height: 100%; border-radius: 4px; width: 0%; transition: width .3s ease; }
    .fill-base  { background: linear-gradient(90deg, #00e5a0, #00b89a); }
    .fill-mejor { background: linear-gradient(90deg, #00b8ff, #007acc); }
    .fill-svm   { background: linear-gradient(90deg, #ff9f43, #e67e22); }
    .fill-rf    { background: linear-gradient(90deg, #a855f7, #7c3aed); }
    .fill-top   { filter: brightness(1.3); }
    .barra-pct { font-size: 9px; color: var(--text-muted); text-align: right; }
    .umbral-note {
      padding: 8px 14px; font-size: 9px; color: var(--text-muted);
      border-top: 1px solid var(--border); line-height: 1.5;
    }
    .umbral-note strong { color: var(--text); }
    footer {
      display: flex; align-items: center; justify-content: space-between;
      padding: 0 14px; background: var(--surface); border-top: 1px solid var(--border);
      font-size: 9px; color: var(--text-muted);
    }
    ::-webkit-scrollbar { width: 4px; }
    ::-webkit-scrollbar-track { background: transparent; }
    ::-webkit-scrollbar-thumb { background: var(--border); border-radius: 2px; }
  </style>
</head>
<body>
<div class="shell">
  <header>
    <div class="logo"><span>●</span> Reconocimiento de Emociones</div>
    <div class="badge">
      <span class="status-dot" id="statusDot"></span>
      <span id="statusText">Conectando...</span>
    </div>
    <div class="badge">UNAM · Mecatrónica · 2026</div>
  </header>
  <main>
    <div class="video-panel">
      <img id="videoFeed" src="/video_feed" alt="Video en tiempo real">
      <div class="fps-badge" id="fpsBadge">— fps</div>
      <div class="no-face-badge" id="noFaceBadge">Sin rostro detectado</div>
    </div>
    <div class="side-panel">
      <div class="section-header">Enfoque basado en imagenes (pixeles)</div>
      <div class="model-card">
        <div class="model-title">CNN Base <span class="model-tag tag-base">3 capas Conv2D</span></div>
        <div class="model-desc">Pixeles grises 96x96 · Haar Cascade · Acc: 98.89%</div>
        <div class="emocion-display">
          <div class="emocion-nombre" id="base-nombre">—</div>
          <div class="emocion-conf conf-alto" id="base-conf">—</div>
```

```
        </div>
        <div class="barras">
          {% for em in emociones %}<div class="barra-row">
            <span class="barra-label">{{ em }}</span>
            <div class="barra-track"><div class="barra-fill fill-base" id="base-fill-{{ em
}}"></div></div>
            <span class="barra-pct" id="base-pct-{{ em }}">0%</span>
          </div>{% endfor %}
        </div>
      </div>
      <div class="model-card">
        <div class="model-title">MobileNetV2 <span class="model-tag tag-mejor">Transfer
Learning</span></div>
        <div class="model-desc">Pixeles RGB 96x96 · ImageNet · Acc: 92.78%</div>
        <div class="emocion-display">
          <div class="emocion-nombre" id="mejor-nombre">—</div>
          <div class="emocion-conf conf-alto-azul" id="mejor-conf">—</div>
        </div>
        <div class="barras">
          {% for em in emociones %}<div class="barra-row">
            <span class="barra-label">{{ em }}</span>
            <div class="barra-track"><div class="barra-fill fill-mejor" id="mejor-fill-{{ em
}}"></div></div>
            <span class="barra-pct" id="mejor-pct-{{ em }}">0%</span>
          </div>{% endfor %}
        </div>
      </div>
      <div class="section-header">Enfoque basado en geometria (MediaPipe)</div>
      <div class="model-card">
        <div class="model-title">SVM <span class="model-tag tag-svm">RBF kernel</span></div>
        <div class="model-desc">11 features geometricas · MediaPipe FaceMesh · Acc: 98.33%</div>
        <div class="emocion-display">
          <div class="emocion-nombre" id="svm-nombre">—</div>
          <div class="emocion-conf conf-alto-naranja" id="svm-conf">—</div>
        </div>
        <div class="barras">
          {% for em in emociones %}<div class="barra-row">
            <span class="barra-label">{{ em }}</span>
            <div class="barra-track"><div class="barra-fill fill-svm" id="svm-fill-{{ em
}}"></div></div>
            <span class="barra-pct" id="svm-pct-{{ em }}">0%</span>
          </div>{% endfor %}
        </div>
      </div>
      <div class="model-card">
        <div class="model-title">Random Forest <span class="model-tag tag-rf">300 arboles</span></div>
        <div class="model-desc">11 features geometricas · MediaPipe FaceMesh · Acc: 99.44%</div>
        <div class="emocion-display">
          <div class="emocion-nombre" id="rf-nombre">—</div>
          <div class="emocion-conf conf-alto-morado" id="rf-conf">—</div>
        </div>
        <div class="barras">
          {% for em in emociones %}<div class="barra-row">
            <span class="barra-label">{{ em }}</span>
            <div class="barra-track"><div class="barra-fill fill-rf" id="rf-fill-{{ em
}}"></div></div>
            <span class="barra-pct" id="rf-pct-{{ em }}">0%</span>
          </div>{% endfor %}
        </div>
      </div>
      <div class="umbral-note">
        <strong>Umbral: 70%</strong> — por debajo = estado incierto.<br>
        <strong>Arriba:</strong> CNNs analizan pixeles (dependen de condiciones).<br>
        <strong>Abajo:</strong> SVM/RF miden geometria facial (mas estables).
      </div>
    </div>
  </main>
  <footer>
    <span>Guerrero Martínez Erick &amp;amp; Salgado Miranda Maya Jily</span>
```

```
    <span id="footerStatus">Iniciando...</span>
    <span>UNAM 2026</span>
  </footer>
</div>
<script>
const EMOCIONES = {{ emociones | tojson }};
const UMBRAL = 0.70;
const MODELS = [
  {key:"cnn_base", prefix:"base",  color:"conf-alto",         ok:"base_ok"},
  {key:"cnn_mejor",prefix:"mejor", color:"conf-alto-azul",    ok:"mejor_ok"},
  {key:"svm",      prefix:"svm",   color:"conf-alto-naranja", ok:"svm_ok"},
  {key:"rf",       prefix:"rf",    color:"conf-alto-morado",  ok:"rf_ok"},
];

function update(prefix, data, colorClass, ok) {
  const n = document.getElementById(`${prefix}-nombre`);
  const c = document.getElementById(`${prefix}-conf`);
  if (!ok || !data) {
    n.textContent = ok ? "—" : "No cargado"; n.style.opacity = ok ? "1" : "0.35";
    c.textContent = "—"; c.className = "emocion-conf conf-bajo";
    EMOCIONES.forEach(e => {
      const f = document.getElementById(`${prefix}-fill-${e}`);
      const p = document.getElementById(`${prefix}-pct-${e}`);
      if(f) f.style.width="0%"; if(p) p.textContent="0%";
    });
    return;
  }
  n.style.opacity = "1"; n.textContent = data.emocion;
  c.textContent = data.confianza > 0 ? `${(data.confianza*100).toFixed(0)}%` : "—";
  c.className = `emocion-conf ${data.confianza >= UMBRAL ? colorClass : "conf-bajo"}`;
  EMOCIONES.forEach(e => {
    const pct = (data.probs[e]||0)*100;
    const f = document.getElementById(`${prefix}-fill-${e}`);
    const p = document.getElementById(`${prefix}-pct-${e}`);
    if(f){ f.style.width=pct.toFixed(1)+"%";
f.classList.toggle("fill-top",e===data.emocion&&data.confianza>=UMBRAL); }
    if(p) p.textContent=pct.toFixed(0)+"%";
  });
}

async function poll() {
  try {
    const r = await fetch("/predictions");
    const d = await r.json();
    const any = MODELS.some(m=>d.modelos[m.ok]);
    document.getElementById("statusDot").className = "status-dot"+(any?" live":"");
    document.getElementById("statusText").textContent = any?(d.rostro?"Rostro detectado":"En
espera"):"Sin modelos";
    document.getElementById("fpsBadge").textContent = `${d.fps} fps`;
    document.getElementById("noFaceBadge").style.display = d.rostro?"none":"block";
    MODELS.forEach(m => update(m.prefix, d[m.key], m.color, d.modelos[m.ok]));
    document.getElementById("footerStatus").textContent = new Date().toLocaleTimeString("es-MX");
  } catch(e) {
    document.getElementById("statusDot").className = "status-dot";
    document.getElementById("statusText").textContent = "Sin conexion";
  }
}
poll(); setInterval(poll, 250);
document.getElementById("videoFeed").addEventListener("error", function(){
  setTimeout(()=>{ this.src="/video_feed?"+Date.now(); }, 2000);
});
</script>
</body>
</html>
```