UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería


# Implementación de un control de movimiento para dispositivos robóticos móviles utilizando identificación de gestos faciales y orientación de cabeza por medio de visión de computadora

Trabajo de graduación presentado por Gerardo Andres Fuentes Bámaca
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería


# Implementación de un control de movimiento para dispositivos robóticos móviles utilizando identificación de gestos faciales y orientación de cabeza por medio de visión de computadora

Trabajo de graduación presentado por Gerardo Andres Fuentes Bámaca
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2024

Vo.Bo.:

(f) _____

PhD. Luis Rivera

Tribunal Examinador:

(f) _____

PhD. Luis Rivera

(f) _____

MSc. Carlos Esquit

(f) _____

Ing. Luis Pedro Montenegro

Fecha de aprobación: Guatemala, 5 de diciembre de 2018.

# Prefacio

# Índice

# Lista de figuras

# Lista de cuadros

# Resumen

En una sociedad donde la tecnología los métodos numéricos y cálculos computacionales superan las habilidades humanas, es necesario aplicar nuestro sentido de la empatía y humanidad para poder generar un verdadero desarrollo humano. Una de las ramas en la que se debe trabajar para lograr este desarrollo, es la inclusividad. Con este tema en mente, se plantea un proyecto que logre unir la visión de computadora y la robótica para generar un prototipo de agente móvil el cual pueda funcionar con base en gestos y movimientos faciales. Para esta tarea será importante lograr desarrollar sistemas efectivos de de software y hardware, así como la comunicación entre ellos.

Para iniciar, se investigará y analizará la disponibilidad de tecnología, tanto software como hardware para implementar visión de computadora a nivel local. Esto se realizará por medio de cotizaciones y análisis de hojas de datos y manuales disponibles, así como la comparación entre plataformas de software. Luego, con los materiales definidos, se procederá a analizar y ajustar los algoritmos y plataformas a utilizar por separado. Después de verificado su correcto funcionamiento se empezará la etapa de desarrollo de algoritmos, en la que se combinarán y probarán las componentes de software, para ser verificadas únicamente por el sensor que se haya seleccionado previamente. Siguiendo esa línea de trabajo, el siguiente paso será verificar la integración del sistema de visión de computadora con simulaciones de agentes robóticos móviles. Cuando los resultados sean satisfactorios, se procederá a migrar este sistema a un agente móvil físico.

# Abstract

# CAPÍTULO 1

## Introducción

# CAPÍTULO 2

# Antecedentes

## Control de gestos para su uso en automóviles

El artículo trata sobre el uso del control de gestos como control para conductores sobre los sistemas integrados en automóviles. Se centra en la detección y reconocimiento de gestos manuales mediante técnicas de recolección y procesamiento de imágenes. En primer lugar se resalta la importancia de tener un contraste entre la mano y el fondo para una detección precisa de gestos. Para lograr esto, los se usó un conjunto de LED que emiten luz infrarroja cercana (NIR) con longitud de onda de 950 nm, la cual no es visible para el ojo humano pero si para una cámara CCD modificada. Esta sensor elimina la información de color pero permite la medición de intensidades. [1]

El proceso de recolección de imágenes implica segmentar cada imagen utilizando una única clase o categoría, por medio de umbralización simple. Luego, se rastrean y etiquetan los límites de las regiones conectadas con características iniciales como tamaño, alcance, centroide y momentos de Hu. Estas características ayudan a identificar y distinguir diferentes gestos manuales. Después, en una secuencia de imágenes con duración preestablecida, se emparejan regiones correspondientes (denominadas objetos) entre imágenes consecutivas. Esto se realiza al rastrear los centroides y asumiendo que las regiones pequeñas representan un ruido casi nulo. Por último, se calculan características de movimiento como velocidad y aceleración. Esto permite clasificar de objetos en diferentes clases dinámicas. Para reducir el peso computacional, se aplica un método de preselección basado en puntuación difusa y así eliminar algunos objetos. Entonces se utilizan conjuntos difusos para etiquetar objetos en una clase que indica que no son manos, basándose en el análisis de rangos típicos de características para manos. También incluye una fase de reconocimiento, donde los objetos se comparan con gestos de referencia precargados. En este caso se generan funciones de distribución de probabilidad para cada característica, y se mide la similitud entre el objeto observado y la referencia. Para evitar clasificaciones falsas, se determina un umbral por defecto y, si el objeto muestra poco movimiento, se usa un segundo clasificador basado en correlación de forma. La tasa de procesamiento se configuró en 25 fotogramas por segundo

con una resolución de 192 × 144 píxeles y una procesador Pentium-II de 333 MHz. [1]

## Reconocimiento de gestos automático para interacciones inteligentes humano-robot

El enfoque principal de este artículo es el reconocimiento de gestos de cuerpo completo para la interacción inteligente entre humanos y robots. Se describe un método basado en aprendizaje que puede realizar simultáneamente la detección y el reconocimiento de gestos importantes. El método implica la estimación de la pose del cuerpo humano en 3D, el entrenamiento de Modelos Ocultos de Markov (HMM, por sus siglas en inglés) para modelar la variabilidad de patrones, y la construcción de un modelo para gestos no realizados. El método propuesto se integra en un sistema robótico llamado T-Rot y logra un alto rendimiento en el reconocimiento. La metodología utilizada fue la siguiente:

### Modelo humano en 3D:

Se construyó una base de datos jerárquica del cuerpo humano utilizando imágenes de silueta e imágenes de profundidad. El modelo incluye múltiples niveles para representar diferentes posturas del cuerpo humano. [2]

### Extracción de características:

Basado en información sobre las componentes del cuerpo en 3D se seleccionan trece puntos característicos. También se utilizaron los ángulos desde el eje vertical hasta cada punto característico como una característica. [2]

### Fase de entrenamiento:

En esta etapa se entrenan los HMM para modelar la variabilidad de los patrones. En este entrenamiento se utilizó la base de datos *KU Gesture* y se aplica el algoritmo K-means para dividir los datos de entrenamiento en sub-categorías. [2]

### Fase de reconocimiento:

En la fase de reconocimiento, se utiliza el algoritmo de Viterbi para encontrar la secuencia de estados más probable para un gesto ingresado al modelo. El gesto se detecta y reconoce en función de los HMMs entrenados. [2]

**Resultados experimentales:**

El método propuesto se evalúa utilizando la base de datos *KU Gesture* y gestos generados. El resultado general de detección es del 94.8 %, y el resultado de reconocimiento de gestos aislados es del 97.4 %. [2]

## Sistema de reconocimiento de manos y rostro para personas ciegas

Este artículo presenta un sistema de reconocimiento diseñado para ayudar a personas con discapacidad visual. Se han desarrollado sistemas de reconocimiento de gestos con la mano y facial para realizar diversas tareas. El sistema captura imágenes dinámicas de un flujo de video, procesándolas a través de algoritmos respectivos. Para el reconocimiento de gestos con la mano, el sistema primero detecta la región de la mano en imágenes en tiempo real. Esto quiere decir que se realiza conversión del espacio RGB al espacio de color YCbCr. El sistema establece valores umbral superiores e inferiores para la detección de piel, los cuales pueden ser ajustados dinámicamente por el usuario según el entorno. Luego, por medio del algoritmo *Convex Hull* o de envoltura convexa el sistema extrae características de la región de la mano, como las puntas de los dedos y el ángulo entre los dedos. En esta aplicación se reconocen diferentes gestos que representan números del uno al cinco utilizando este sistema. Para el reconocimiento facial, el sistema utiliza clasificadores de cascada Haar y el reconocedor LBPH (Histogramas de Patrones Binarios Locales). La imagen capturada del rostro se convierte a una imagen en escala de grises. Luego, el sistema entrena la base de datos de imágenes utilizando clasificadores y reconocedores. Durante el reconocimiento, la imagen en tiempo real se compara con las imágenes en la base de datos. Si se encuentra una coincidencia, se muestra el nombre asociado con la imagen. [3]

## Sistema de reconocimiento gestos de cabeza utilizando la cámara de gestos

La Cámara de gestos es una cámara inteligente diseñada para capturar e interpretar gestos de cabeza, aí como expresiones faciales. Esta enfocado principalmente en individuos con discapacidades o parálisis. El documento explora diferentes áreas, como reconocimiento de gestos, detección facial, rastreo de rostro, y detección de obstáculos. Este sistema cuenta con unidades de captura de imágenes, reconocimiento de gestos y la de concentración despliegue de datos. La unidad de captura de imágenes utiliza un sensor de imagen a color CMOS, montada en una placa electrónica. Luego de capturar la secuencia de imágenes o video, la unidad de reconocimiento de gestos analiza e identifica los gestos y expresiones faciales. Para poder realizar esto, utiliza técnicas como modelado de movimiento, análisis de movimiento, reconocimiento de patrones, y *machine learning* para poder interpretar los gestos. Esta unidad también es capaz de detectar el movimiento de la cabeza par luego determinar su orientación. Una vez que los gestos se reconocieron, la información se transfiere a la unidad de concentración y despliegue. Esta unidad puede ser una computadora personal

o bien una de control centralizado. La salida de datos de la cámara inteligente, que incluye una descripción de alto nivel del rostro del usuario en diferentes orientaciones, es enviada a la sección de concentración para un futuro procesamiento. [4]

- Los retos mas importantes afrontados en el desarrollo de dicho sistema fueron:

    1. Imágenes fuera del rango de la cámara, esto fue un problema dado que el sistema debe ser capaz de detectar los gestos de movimiento incluso si la cabeza no es completamente visible por el sensor. [4]
    2. La variación de las condiciones de iluminación, el sistema también debe ser capaz de reconocer los gestos de la cabeza con precisión aún si las condiciones de luz cambian, entendiendo que esto puede cambiar el aspecto del rostro del usuario. [4]
    3. La variación de las formas de los rostros, los usuarios jamás tendrán rostros exactamente iguales, tanto por forma, tono de piel, bello facial, gafas, entre otros, dicha variación provoca un mayor reto para reconocer gestos de manera adecuada. [4]
    4. Fondos desordenados, cuando el dispositivo móvil realice una acción de movimiento, el fondo puede parecer desordenado o en movimiento, lo cual se puede mezclar con el movimiento de los gestos, creando así una dificultad para diferenciar el fondo de los gestos en el algoritmo de reconocimiento del sistema. [4]

## Control de una silla de ruedas usando una agrupación de k-medias adaptativas de las poses de la cabeza

En este artículo se presenta la idea de ayudar a las personas con ciertas discapacidades físicas, por modio de una interfaz que utiliza el sensor Kinect desarrollado por Microsoft. El objetivo es permitir a las personas con discapacidades interactuar con los dispositivos electrónicos. Para lograr esto, se utilizó un algoritmo basado en la Regresión de Bosques Aleatorio y uno de agrupación de k-medias para detectar los cambios en el rango de dirección de los movimiento de cabeza. Y la experimentación se realizó en 5 individuos operando la silla de ruedas en condiciones favorables y no favorables. [5]

El sistema propuesto se basa en el método de agrupamiento de k-medias, el cual es un esquema de agrupamiento que utiliza puntos representativos y distancias euclidianas para medir escala de similitud entre vectores y el agrupamiento de puntos representativos. Además los datos obtenidos son posibles gracias al sensor de profundidad utilizado. La arquitectura del sistema propuesto funciona de la siguiente manera:

### Calibración

Este proceso consta del establecimiento de la configuración inicial para cada usuario, del tal manera que se pueda realizar un proceso personalizado al usuario. Este proceso requiere que el usuario coloque la posición de la cabeza según solicite la interfaz, para

configurar las funciones de detenerse, izquierda, derecha, adelante y atrás. Este proceso utiliza el algoritmo RRF. Luego, los datos recopilados se guardan para luego utilizarse en el proceso de adaptación. [5]

### Rectificado

Esta es la etapa final, pero el artículo lo menciona que es importante mencionar en este punto la funcionalidad del proceso. Dado que el sistema se encuentra generando constantemente estimados de los ángulos de pose, es importante realizar un rectificado según la pose y la capacidad de movimiento del usuario. [5]

### Adaptación

Se consideró que, dadas las situaciones de fatiga o degeneración en las enfermedades de los usuarios, podría suceder una descalibración del control de movimiento. Para esto se integraron dos métodos al sistema. El primer método consisten en la implementación de sensores de sonar que se encargan de detectar obstáculos. Al detectarlos, se implementa una corrección en el movimiento. El siguiente método consiste en crear agrupaciones adicionales a las de las funciones principales. Dichas agrupaciones, son las combinaciones de los movimientos básicos. Esto es realizado por medio de el algoritmo de agrupamiento de k-medias, y ayuda a no realizar cambios de movimiento abruptos. [5]

# CAPÍTULO 3

## Justificación

Para llegar a ser una sociedad más inclusiva es necesario poder desarrollar tecnología que afronte diferentes situaciones, en este caso las personas con algún tipo de discapacidad. Existe una gran variedad de extremidades inhabilitadas en el espectro de las discapacidades. En muchos casos bastaría con un sistema de reconocimiento de gestos de manos para personas con parálisis de la cintura para abajo. Sin embargo, también hay que tomar en cuenta a las personas con discapacidades en las extremidades superiores, es ahí donde entra el reconocimiento de gestos de la cabeza. El desarrollo de un sistema de visión de computadora capaz de reconocer gestos faciales/de cabeza, que sea capaz de enviar comandos a un robot móvil es el primer paso para desarrollar una tecnología de apoyo. El control de un robot móvil por medio de este algoritmo, dará paso a una futura migración de comandos a plataformas móviles como sillas de ruedas, o bien vehículos eléctricos. Esta alternativa también busca utilizar sensores mucho más comunes en el mercado, como lo son los sensores de imágenes a comparación de sensores de profundidad o bien de señales bioeléctricas. Los sensores de imágenes, además tienen el valor agregado de brindar una mayor libertad de movimiento a los usuarios, ya que no son intrusivos, por lo que el gesto realizado tiende a una mayor naturalidad. Esto sucede dado que al utilizar aquellos que funcionan con EEG o EMG, requieren dispositivos de extensión al cuerpo humano, como son gorras, guantes o electrodos adheridos al cuerpo del sujeto de prueba. Por otro lado, también se espera que la sección de corrección por integración de sensores, permita una mayor seguridad al usuario. [6], [4]

CAPÍTULO 4

# Objetivos

## Objetivo general

Desarrollar un sistema de visión por computadora para reconocimiento de gestos y orientación de la cabeza para el control de un agente robótico móvil.

## Objetivos específicos

- Investigar y seleccionar el hardware y software disponible para la aplicación de visión por computadora.
- Implementar un algoritmo de visión por computadora para reconocimiento de gestos y movimiento de la cabeza.
- Implementar un algoritmo de traducción de los gestos y movimientos de la cabeza a comandos de control para el agente robótico.
- Validar el correcto funcionamiento del sistema de reconocimiento y control por medio de simulaciones computarizadas.
- Validar el correcto funcionamiento del sistema de reconocimiento y control por medio de plataformas robóticas móviles.

# CAPÍTULO 5

## Alcance

# CAPÍTULO 6

# Marco teórico

## Visión por computadora

Como humanos percibimos el mundo tridimensional con bastante facilidad, esto es ejemplificado al momento de hacer la diferenciación entre los detalles de macetas o bien contar personas o interpretar sus emociones en un retrato grupal. Replicar este proceso es una tarea que tanto los psicólogos e investigadores de visión por computadora han intentado por años. Mientras que las ilusiones ópticas nos ofrecen algunos indicios, el entendimiento por completo aún es incierto. En el mundo de la visión por computadora, se han hecho avances en técnicas matemáticas para reconstruir formas e imágenes tridimensionales, el seguimiento de objetos, e incluso el reconocimiento de personas en fotografías. Sin embargo, alcanzar una interpretación de imágenes comparable a la de niños pequeños aún es un reto. La visión posee dificultades dado que es un problema de ingeniería inversa, el cual requiere modelos basados en física y estadística para navegar en la ambigüedad. Actualmente, la visión por computadora tienen sus cimientos en dichos modelos de física y gráficas de computadora, abordando problemas sobre cómo se mueven los objetos, el reflejo de la luz e interacción con el ambiente. Hoy, la visión de computadora es subestimada por personas externas debido a una malentendido con respecto al funcionamiento, esta es una creencia refutada, esta afirmaba que la percepción era más simple que el pensamiento cognitivo. En la actualidad, existen muchas aplicaciones de visión por computadora, las cuales tienen diferentes retos respecto de su complejidad en matemáticas, o de la naturaleza misma del problema. En visión de computadora, es sumamente importante aplicar métodos o técnicas que se acoplen al problema o reto en vez de producir métodos genéricos. De esta manera se genera un pensamiento ingenieril para la resolución de problemas, dado que muchas veces se requiere modelos de la física del caso y así como la estadística para modelar escenarios y el ruido que se pueda obtener un escenario más fiel a la realidad. [7]

El campo de la visión por computadora es una ramificación de la inteligencia artificial, la cual tiene el propósito de utilizar computadoras y sistemas para poder obtener información importante desde imágenes, videos u otros datos visuales, para luego poder realizar acciones.

Este campo es un sistema que se forma con componentes de *hardware*, los cuales pueden ser cámaras, iluminación, lentes, sensores de imágenes, etc. Además, también deben poseer una sección de *software*, que se refiere a los algoritmos para procesar y analizar imágenes, videos, etc. También, es necesario tomar en cuenta la comunicación entre el *hardware* y *software*, dado que esto es lo que permite el funcionamiento del sistema, para esto existen diferentes protocolos. [8]

## Procedimientos

### Clasificación de imágenes:

Es el proceso de categorizar o etiquetar imágenes en diferentes clases o cateogrías predefinidas. Esto incluye el entrenamiento de un modelo de *Machine Learning* para reconocer y diferenciar entre los diferentes objetos, escenarios, o patrones de imágenes. El objetivo es desarrollar un modelo que pueda clasificar de manera eficaz imágenes nuevas o no antes vistas, basado en los patrones y características que se han aprendido previamente. La clasificación de imágenes tiene muchos campos, como detección de objetos, detección de rostros, imágenes médicas, y vehículos autónomos. El proceso para realizar la clasificación de imágenes sigue de la siguiente manera:

**Recolección de datos:** En esta etapa se debe recolectar un grupo grande de imágenes las cuales estén previamente etiquetadas en clases o categorías específicas.[9]

**Procesamiento de datos** Esta parte se encarga de limpiar y realizar un proceso previo a las imágenes de tal manera que aseguren que tienen el formato adecuado para ingresar al modelo. Esta etapa incluye, reajuste de tamaño o proceso de normalizado. [9]

**Extracción de características:** En este caso se extraen características únicas de las imágenes que ayuden a distinguir entre clases, esto se puede hacer con diferentes técnicas como redes neuronales convolucionales o con otro métodos menos automáticos. [9]

**Entrenamiento del modelo:** Para poder evaluar el modelo, se necesita un grupo de datos separado, para verificar el rendimiento. Las métricas de evaluación normalmente son: exactitud, precisión, *recall*, y la calificación F1. [9]

**Optimización del modelo:** Ajustar el modelo es posible utilizando los hiperparámetros, como la optimización del proceso de entrenamiento, o usando técnicas como la regularización.[9]

**Predicción:** Ya que el modelo está entrenado y optimizado, se puede usar para clasificar nuevos datos, o datos no antes vistos, y este devuelve la categoría o clase en la que fue clasificado el dato o imagen. [9]

**Detección de objetos:**

La detección de objetos es una técnica de visión por computadora para localizar instancias de objetos en imágenes o videos. Los algoritmos de detección de objetos suelen aprovechar el aprendizaje automático o el aprendizaje profundo para producir resultados significativos. La detección de objetos es una tecnología clave detrás de los sistemas avanzados de asistencia al conductor (ADAS) que permiten a los automóviles detectar carriles de conducción o realizar la detección de peatones. La detección de objetos también es útil en aplicaciones como sistemas de vigilancia de video o sistemas de recuperación de imágenes. [10]

**Estimación de pose:**

Es el proceso de determinar la posición y orientación de un objeto en un sistema de coordenadas. Se usa comúnmente en visión de computadora y aplicaciones robóticas. En el contexto de estimación de pose de cámaras, el objetivo es determinar la posición y orientación de una cámara relativa al escenario tridimensional. Esto se realiza normalmente, por medio del emparejamiento de los puntos en imágenes 2D con sus puntos correspondientes en el espacio 3D. Este proceso incluye resolver el problema de la Perspectiva en punto o *Perspective-n-Point*, el cual implica encontrar la pose de una cámara dado un grupo de puntos correspondientes entre el plano 2D y el espacio 3D. Para realizar la estimación de pose existen muchos métodos, incluyendo iterativos y no iterativos. Los métodos iterativos realizan una refinación del estado inicial por medio de un proceso de estimación. Por otro lado, los métodos no iterativos calculan la pose una única vez, sin iteraciones. Un método popular no iterativo es el algoritmo EPnP (*Efficiente Perspective-n-Point*), que está basado en la resolución de un grupo de ecuaciones lineales. Esta utiliza una suma ponderada de puntos de control virtuales para representar puntos en tres dimensiones y reducir el problema a la estimación de las coordenadas de dichos puntos. Este método es computacionalmente eficiente y devuelve resultados precisos. [11]

**Rastreo de objetos:**

En visión de computadora el rastreo de datos involucra la aplicación de algoritmos para detectar y seguir el movimiento de objetos específicos dentro de un video o a lo largo de cuadros. Este proceso es esencial para varias aplicaciones, desde el rastreo de robots en bodegas hasta el rastreo en sistemas de drones. El rastreo tiene sus pilares en la detección de objetos, sin embargo el objeto puede tener distintas apariencias dependiendo de los escenarios y los ángulos. El algoritmo de rastreo predice la posición de un objeto en cuadros de imagen consecutivos, al mismo tiempo que identifica y sigue más de un objeto en una imagen o video. El proceso de rastreo de objetos inicia con la definición del objeto de interés, esto se hace encerrando al objeto en un recuadro, en el primer cuadro del video. El algoritmo de rastreo, luego estima o predice la posición en los cuadros restantes mientras vuelve a encerrar al objeto en ese recuadro de manera simultánea. El algoritmo para rastreo de objetos puede clasificarse basado en el número de objetos que rastrean, incluyendo el rastreo de un único objeto, el cual implica rastrear un objetivo a la vez, y el rastreo múltiple de

objetos. Los algoritmos de rastreo tienen muchos retos como ambientes complejos, y la aplicación de diferentes técnicas, como el aprendizaje profundo. El rastro de objetos es una tarea importante en la visión de computadora por sus aplicaciones en, monitoreo de tráfico, robótica, imágenes médicas, etc. [12] [13] [14] [15]

#### Reconocimiento de gestos:

La interacción entre computadoras y humanos existe de diferentes maneras, pero la interacción por medio de gestos tiene un pesos importante dado que, le permite a los humanos una mayor naturalidad al momento de comunicarse. Hoy, existen diferentes métodos para reconocimiento de gestos, los cuales requieren dispositivos que sirven como extensión al cuerpo humano. Sin embargo, estos limitan la naturalidad con que un sujeto de prueba puede interactuar con su ambiente. La alternativa que se propone, desde hace más de veinte años, es la visión de computadora, dado que para capturar los gestos de la persona únicamente se requiere una cámara . Actualmente, el reconocimiento de gestos por visión de computadora está basado en su mayor parte en el *Hidden Markov Model (HMM)*, los algoritmos de redes neuronales y el algoritmo de redondeo del tiempo dinámico. Los pasos para realizar dicho reconocimiento, son: recopilación de imágenes, segmentación de manos, reconocimiento de gestos y clasificación. [6]

La recolección de imágenes se divide en dos partes fundamentales, RGB y de profundidad. Comúnmente, se utiliza únicamente cámaras para detectar el RGB, sin embargo, para la profundidad es necesario utilizar otro tipo de sensores, los cuales puedan devolver información de profundidad. Además, para poder analizar correctamente los gestos es necesario también dividirlos entre estáticos y dinámicos. Dicha división se refiere a la distinción entre fotos o videos respectivamente. Lastimosamente, el proceso de recolección de imágenes se ve afectado por problemas de dirección e intensidad de luz, por lo que los algoritmos realizados deben apuntar a ser invariantes respecto iluminación. [6]

### Hardware

#### Cámaras o sensores de imágenes:

Los sensores de las cámaras digitales son dispositivos electrónicos que contienen millones de píxeles fotosensibles, estos registran la cantidad de luz que reciben. Al presionar el botón para tomar la fotografía, se descubren los píxeles para recoger los fotones y almacenarlos como una carga eléctrica. Cuando termina la exposición a la luz, la cámara cierra cada uno de estos píxeles y luego revisa cuántos fotones cayeron, midiendo así, la intensidad de la señal eléctrica. Estos datos se guardan como valores digitales, y su precisión es determinada por la profundidad de bits. Luego, los datos se procesan y convierten en una imagen que se guarda en una tarjeta de memoria o en la cámara por sí misma. En la actualidad, estos sensores se utilizan en cámaras digitales, módulos de cámara, teléfonos con cámara, equipos de imágenes médicas, dispositivos de visión nocturna, etc. En productos de consumo diario normalmente utilizan sensores *Semiconductores complementarios de óxido metálido* o CMOS, por sus siglas en inglés, los cuales son en su mayoría más baratos y su consumo de energía es bajo en dispositivos con batería. Por otro lado, los sensores *Dispositivos de carga acoplada* o CCD,

por sus siglas en inglés, se utilizan en cámaras de video de alta calidad. Ambos tipos de sensores cumplen la misma tarea de capturar luz y convertirla en señales eléctricas. Los sensores de imagen también pueden tener diferentes tamaños y resoluciones, lo cual afecta la calidad de la imagen. Los sensores más grandes tienen una mayor capacidad para capturar luz y, por ende, pueden producir imágenes de mayor calidad y con menor ruido. Además, los sensores de imagen también pueden tener diferentes formatos, como el *Full-frame*, el formato APS-C y el formato *Micro Four Thirds*, que afectan el ángulo de visión y la profundidad de campo. [16]

**Sensor LiDAR:**

LiDAR, o *Light Detection and Ranging*, es una tecnología de teledetección, y funciona al emitir pulsaciones láser y luego midiendo el tiempo que tarda la luz en regresar después de golpear un objeto o la superficie terrestre. Al medir el tiempo de los pulsos con precisión, los sistemas LiDAR pueden calcular la distancia al objeto o superficie. Este proceso se repite varias veces para crear un mapa tridimensional detallado. Los datos resultantes suelen representarse como un sistema de coordenadas tridimensional. En los vehículos autónomos, LiDAR es un componente importante que les permite percibir y navegar su entorno con alta precisión y exactitud. En robótica, se utiliza para la localización y mapeo simultáneos, lo que permite a los robots crear mapas de su entorno y navegar dentro de él. También, se usa para crear mapas altamente detallados del terreno, vegetación y cobertura terrestre, lo cual ayuda a la gestión ambiental y de agricultura. En la planificación urbana, los datos se utilizan para crear modelos tridimensionales detallados de ciudades, par evaluar infraestructuras. Las ventajas de este sistema incluyen: datos tridimensionales altamente precisos y detallados, que lo hace valioso para diversas aplicaciones que requieren información espacial. También, puede utilizarse en diversos entornos y para muchas aplicaciones, como la gestión ambiental y desarrollo de infraestructuras mencionadas anteriormente. Además tiene la capacidad de capturar rápidamente grandes cantidades de datos, lo que permite un eficiente mapeo y topografía de áreas extensas. Por otro lado, sus mayores desventajas radican en el alto costo y en la baja compatibilidad de los datos recopilados. [17]

**Sensor RADAR:**

Estos dispositivos convierten las señales de eco de microondas en señales eléctricas, utilizando tecnología inalámbrica para detectar el movimiento por medio de la posición, forma, tipo de movimiento y trayectoria de un objeto. A diferencia de otros sensores, no se ven afectados por el espectro de luz, y tienen la capacidad de detección aunque existan obstáculos de por medio. Una de sus principales ventajas es su capacidad para detectar el movimiento al calcular la velocidad y dirección de un objeto a través del efecto Doppler, así como observar el movimiento del objetivo desde diferentes perspectivas usando sensores multicanal. Los sensores de radar se utilizan cada vez más en dispositivos portátiles, edificios inteligentes y vehículos autónomos, y su importancia histórica es evidente en aplicaciones como la predicción del clima y el seguimiento de la temperatura. [18]

Estos sensores funcionan detectando la radiación electromagnética transmitida y reflejada. Envían pulsos electromagnéticos cronometrados y analizan la diferencia entre las señales

enviadas y recibidas para detectar el movimiento, la presencia, la distancia, la velocidad y la localización de objetos. Estos sensores se utilizan en una gran variedad de campos donde se requiere una acción automática al detectar un objeto en movimiento, como controlar luces, sistemas de calefacción y refrigeración, dispositivos de encendido/apagado y abrir o cerrar puertas. Además, uno de los objetivos clave al utilizar sensores de radar es reducir el consumo de energía. [19]

**Cámaras *Time-of-Flight (TOF)*:**

Una cámara con esta tecnología opera al iluminar el espacio o escena con una fuente de luz modulada, normalmente con un láser de estado sólido o un LED operando cerca del rango infrarrojo, por ende la luz es invisible al ojo humano. Luego, la cámara percibe la luz reflejada y mide el desfase entre la ilumiinación y la reflección. Este desfase es usado para determinar la distancia entre objetos dentro del espacio. Las cámaras del TOF utilizan arreglos de píxeles de CMOS de bajo costo y una fuente de luz activa y modulada. La luz que entra a la cámara tiene tanto un componente ambiental como un componente de reflejado. La información de distancia está embebida en el componente reflejado de la luz. Sin embargo, un componente de ambiente robusto puede reducir la razón de señal-ruido de la cámara. Para poder detectar el desfase, la la fuente de luz es modulada de manera continua por medio de una fuente de onda continua, como lo es un sinusoide o bien una señal cuadrada. La modulación por señal cuadrada es más común, dado que se puede realizar fácilmente utilizando circuitos digitales. También, puede ser generada al integrar los fotoelectrones desde la componente de luz reflejada o usando un contador rápido que sea activado por la primera detección de reflexión. Las cámaras con tecnología TOF también incluyen un controlador (TFC), que es una máquina de estados que sincroniza la operación del sensor, la interfaz analógica, y la iluminación. El controlador escanea los pixeles, calcula la profundidad para cada uno, y realiza varias tareas de procesado como eliminación de aliasing, de ruido, afinación de frecuencias, y compensación de temperatura. También maneja la alta tasa de entrada dy salida, serialización y deserialización. [20]

**Kinect:**

El sensor Kinect, es un sensor de profundidad que opera con base en el principio de luz estructurada y aprendizaje automático. Consiste en una cámara infrarroja y un emisor de luz, que proyecta un patrón conocido de luz infrarroja sobre el espacio. El sensor captura la deformación de este patrón, que luego se utiliza para calcular el mapa de profundidad de los objetos en el espacio. Este proceso de cálculo del mapa de profundidad implica analizar el patrón punteado de la luz láser infrarroja. Esta técnica de análisis de un patrón conocido se llama luz estructurada. El principio general es proyectar un patrón conocido sobre el espacio e inferir la profundidad a partir de la deformación de ese patrón. El mapa de profundidad se construye analizando la deformación del patrón de punteado de la luz láser infrarroja. El cálculo de la profundidad se realiza mediante el hardware *PrimeSense* incorporado en Kinect, que utiliza triangulación para calcular el mapa de profundidad. La segunda etapa del proceso implica inferir la posición del cuerpo utilizando el aprendizaje automático. Esta etapa perfecciona el mapa de profundidad obtenido del análisis de luz estructurada, teniendo en cuenta la perspectiva y otros factores para mejorar la precisión de la información de

profundidad. El sensor Kinect ha sido evaluado para diversas aplicaciones de visión por computadora, incluyendo resolución y precisión 3D, ruido estructural, configuraciones de múltiples cámaras y respuesta transitoria del sensor. [21]

### Software:

#### OpenCV:

OpenCV *(Open Source Computer Vision)* es una biblioteca de funciones de programación que se utiliza principalmente para el procesamiento de imágenes. Contiene una API estándar (Interfaz de Programación de Aplicaciones) para aplicaciones de visión por computadora. Al inicio, OpenCV fue desarrollado como un proyecto de investigación de Intel, pero ahora está disponible de manera gratuita bajo la licencia de Distribución de Software de Berkeley. Está escrito principalmente en C++, pero también se puede programar en Python, Java y MATLAB. Es compatible con varias plataformas, incluyendo Windows, Android, iOS, Linux y más. Además, está en constante evolución, con investigación y desarrollo continuos. El propósito principal de OpenCV es ayudar a las computadoras a entender el contenido de las imágenes, por lo que contiene una gran variedad de herramientas y algoritmos que pueden utilizarse para resolver diversos problemas de visión por computadora. Algunas de las herramientas son:

**Filtrado de imágenes:** Contiene funciones para modificar o mejorar imágenes con técnicas como filtrado de imágenes lineales y no lineales. Esto se puede utilizar para eliminar ruido, desenfocar o enfocar imágenes entre otras. [22]

**Transformación de imágenes:** También tiene métodos para generar nuevas imágenes a partir de varias fuentes, destacando características o propiedades específicas. Esto incluye técnicas como la Transformada de Hough para encontrar líneas en una imagen, la Transformada de Radón para reconstruir imágenes a partir de datos de proyección y otras transformadas para compresión de imágenes y videos. [22]

**Detección de características:** Existen herramientas para encontrar características específicas en una imagen, como líneas, bordes o ángulos. Estas características se pueden utilizar como base para otros algoritmos de visión por computadora y son importantes para tareas como el reconocimiento y rastreo de objetos.[22]

**Rastreo de objetos:** También incluye algoritmos para localizar y rastrear objetos en una secuencia de imágenes. Esto es útil en aplicaciones como vigilancia, interacción humano-computadora e imágenes médicas.[22]

**Detección y reconocimiento de rostros:** OpenCV posee modelos y algoritmos preentrenados para detectar y reconocer rostros en imágenes y videos. Esto se puede usar en

aplicaciones como autenticación facial, donde se compara una imagen capturada con una base de datos de rostros conocidos.[22]

**TensorFlow:**

Este es un sistema para aprendizaje automático a gran escala, el cual fue desarrollado Google. Está diseñado para soportar el entrenamiento e inferencia de modelos de aprendizaje automático en diferentes de plataformas, desde clústeres hasta dispositivos móviles. TensorFlow utiliza una representación conceptual de flujo de datos para representar tanto la computación en un algoritmo como el estado en el que se encuentra operando el algoritmo. También, ofrece un modelo de programación flexible y general para la investigación en aprendizaje automático. Este sistema ha sido ampliamente utilizado en producción para Google y también fue lanzado como un proyecto de código abierto. TensorFlow puede utilizarse en diversas aplicaciones, incluyendo la visión por computadora al proporcionar un marco de trabajo robusto para construir y entrenar modelos de aprendizaje profundo, los cuales se utilizan ampliamente en visión por computadora. Además, se puede implementar y experimentar con algoritmos de visión por computadora de última generación, entrenar modelos con conjuntos de datos extensos y utilizarlos para clasificación de imágenes, detección de objetos, segmentación de imágenes, etc. [23]

**CUDA:**

El software NVIDIA CUDA es un modelo de programación y plataforma que permite a los desarrolladores aprovechar las tarjetas gráficas de NVIDIA para diversas aplicaciones. Se basa en una arquitectura de cómputo paralelo y proporciona un modelo de programación que demuestra eficientemente dicho paralelismo de datos. El software permite escribir código que puede ejecutarse en los procesadores paralelos de las GPU de NVIDIA, esto permite procesos de alto rendimiento y la aceleración de diversas aplicaciones. Para visión de computadora el software NVIDIA CUDA es considerado relevante ya que permite aprovechar el paralelismo al analizar datos. CUDA permite el trabajar de manera eficiente grandes cantidades de datos visuales, como procesamiento de imágenes y videos, detección y rastreo de objetos. Al utilizar CUDA, se puede acelerar el rendimiento de algoritmos de visión por computadora, por medio del paralelismo. Una desventaja para esta aplicación, es que requiere una tarjeta gráfica de NVIDIA, que no siempre se encuentra disponible. [24]

**MATLAB:**

Es una plataforma de computación numérica y programación utilizada para analizar datos, desarrollo de algoritmos y creación de modelos. Utiliza un lenguaje de programación que expresa matemáticas de matrices y arreglos directamente. MATLAB cuenta con *Toolboxes* que están desarrolladas profesionalmente, probadas y documentada, además sus aplicaciones permiten ver cómo funcionan diferentes algoritmos. También cuenta con Simulink para aplicar el Diseño Basado en Modelos, que se usa para la simulación multidominio, la generación automática de código y la prueba y verificación de sistemas integrados. En visión de computadora MATLAB ofrece el *Toolbox* de Procesamiento de Imágenes, el *Toolbox* de Visión por

Computadora y el *Toolbox* de LIDAR, además de un conjunto de aplicaciones, algoritmos y redes pre-entrenadas. El *Toolbox* de Visión por Computadora ayuda a identificar errores y defectos en objetos como partes de máquinas, utilizando algoritmos de preprocesamiento de imágenes del *Toolbox* de Procesamiento de Imágenes para mejorar las características. [25]

Las técnicas de aprendizaje profundo normalmente se usan para la detectar defectos o fallas, por medio de aplicaciones como el etiquetador de imágenes, videos o LIDAR de MATLAB, permitiendo la creación de máscaras de segmentación para datos de entrenamiento. La detección y rastreo de objetos, una aplicación esencial en visión por computadora, se puede lograr utilizando el Diseñador de Redes Profundas de MATLAB. Además, dentro del *Toolbox* de Visión por Computadora se puede realizar tareas como localización, mapeo mediante visual SLAM, modelización 3D de objetos a través de SfM y conteo de objetos. Las aplicaciones de MATLAB, como *Image Segmenter* e *Image Region Analyzer*, ofrecen una interfaz interactiva para la segmentación y el conteo de objetos en imágenes, lo que lo hace de fácil acceso. [25]

**CAFFE:**

Es un marco de trabajo o *framework* de aprendizaje profundo diseñado con el objetivo de lograr expresión, velocidad y modularidad. Fue desarrollado por Yangqing Jia y contribuyentes de la comunidad, durante su doctorado en UC Berkeley, bajo el *Berkeley AI Research (BAIR)*. En este marco de trabajo los modelos y la optimización se definen mediante configuraciones, de tal manera que no es indispensable codificar. Además, permite cambiar entre CPU y GPU configurando un solo indicador para entrenar una máquina GPU y luego implementar en clústeres o dispositivos móviles. Dada la sencillez de su configuración Caffe se vuelve perfecto para experimentos de investigación e implementaciones industriales. También, tiene la capacidad para procesar más de 60 millones de imágenes al día con una sola GPU NVIDIA K40 en determinadas condiciones. Lo anterior implica 1 ms/imagen para inferencia y 4 ms/imagen para aprendizaje. [26]

# CAPÍTULO 7

## Selección de hardware y software

## 7.1. Lenguaje de programación

Inicialmente se inicia a trabajar utilizando herramientas presentadas en cursos de robótica, en este caso, se utiliza Python3.10 como lenguaje de programación. Dicha selección, resulta de la compatibilidad que tiene con *OpenCV*, una librería utilizada para poder realizar aplicaciones de visión por computadora, que funciona tanto para leer las imágenes de entrenamiento a utilizar en el modelo de *Machine Learning* como para la captura de imágenes en tiempo real. Es importante mencionar que, para realizar la programación, se optó por utilizar el IDE (Entorno de desarrollo integrado) *PyCharm Community Edition*, dado que cuenta con comandos integrados para manejar entornos virtuales y conexiones con *Git* y *Github*. Además *PyCharm* es uno de los IDE's que cuenta con una fácil incorporación al simulador *Webots*,

## 7.2. Simulador

En este caso se optó por utilizar un software de código abierto, de igual manera que la librería *OpenCV*. Este simulador es conocido como *Webots*, este simulador cuenta con diferentes opciones de ambientes y objetos para personalizar. Dicha personalización se refiere a las características espaciales y físicas, además de contar con diferentes agentes robóticos previamente incluidos. Dichos agentes, poseen el modelo virtual, así como la capacidad de controlarlos por medio de controladores en diferentes lenguajes de programación: C, C++, Java, Python, etc.

## 7.3. Sensor óptico

### 7.3.1. Cámara integrada en laptop

Como primer recurso se tomó en cuenta el uso de la cámara incorporada en una laptop, dado que cualquier persona que esté trabajando en una, puede utilizar dicha herramienta, ahorrando así la compra de un módulo externo. Dicho sistema de cámara integrada puede variar su resolución desde los 0.4 (848×480) Megapíxeles hasta los 2.1(1920×1080). Este módulo se encuentra integrado en una laptop *Lenovo IdeaPad Flex 5 16IAU7* y es fabricado por la empresa *SunplusIT*, la cual es una empresa líder en la manufactura y distribución de chips para multimedia y aplicaciones automotrices, así como el proveedor de cámaras integradas para la empresa *Lenovo*, por lo cual la replicabilidad de las secciones realizadas con dicho sensor es alta.

### 7.3.2. Posible módulo externo

## 7.4. Transmisión de datos

## 7.5. Agente robótico

# CAPÍTULO 8

# Bases de datos

## 8.1. ICPR

Al inicio se realizó una búsqueda de bases de datos que tuvieran diferentes rostros posicionados en distintos ángulos. Sin embargo, hubo una única base de datos pública publicada por la *ICPR (International Workshop on Visual Observation of Deictic Gestures)*, en Cambridge, Reino Unido, con 2790 imágenes disponibles para cualquier propósito. En esta base de datos cuenta con 15 sujetos de prueba, a cada sujeto se le realizaron dos pruebas y en cada prueba se realizó un total de 93 fotografías. El grupo de imágenes se obtuvieron al mezclar los siguientes ángulos: 7 de inclinación (vertical) y 13 panorámicos (horizontal) además de una dos fotografías extra, una en la posición completamente hacia arriba y otra completamente hacia abajo. Las imágenes en esta base de datos se encuentran en un formato JPG, además cada imagen cuenta con un documento de texto que contiene: el nombre del archivo y el centro en el eje X, en el eje Y, el alto y el ancho de la cara correspondiente. [27]

Las fotografías de la base de datos se encuentran nombradas en primer lugar por su ángulo de panorámica y luego por su ángulo de inclinación, por lo que se puede observar que para cada sujeto las fotografías aparecen desde la panorámica extrema derecha y la inclinación extrema inferior y terminan en los extremos opuestos. Por lo que la fotografías recorren primero los ángulos de panorámica y luego incrementan los ángulos de inclinación. Además, estas fotografías no cuentan con algún archivo de etiquetas, por lo que estas se asignaron manualmente, tomando en cuenta el orden previamente explicado en que se encuentran ordenadas. A continuación se muestra un ejemplo del orden mencionado anteriormente:


Figura 1: Muestra de 15 fotografías del sujeto 12 [27]

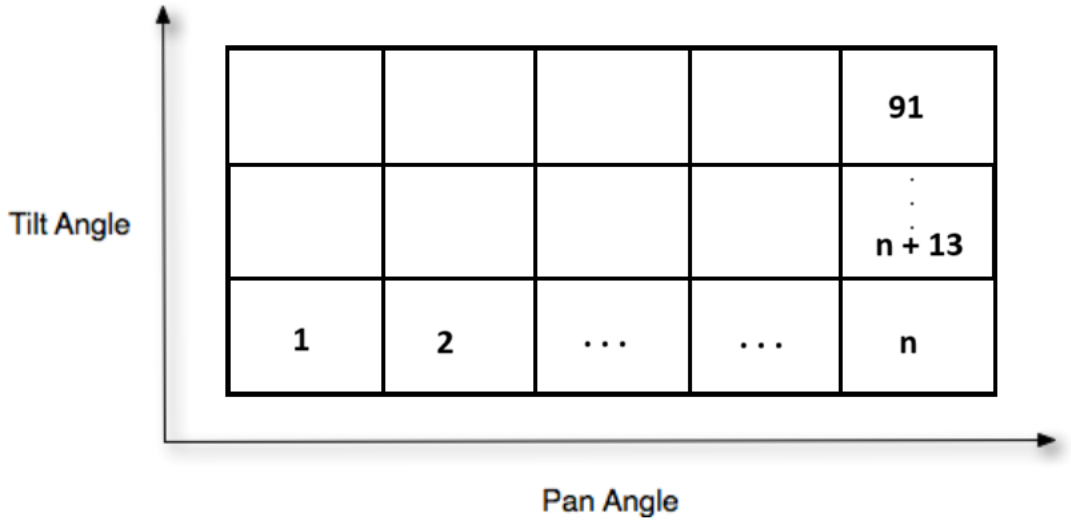


Figura 2: Orden de guardado para cada serie de fotografías

Además de este orden, todas las series de fotografías iniciaban con la imagen de la cabeza completamente hacia arriba y finalizaban con la posición completamente abajo, de frente para ambas. Dado que se tenían 15 sujetos de prueba, se procedió a guardar los archivos JPG de 11 (2046 imágenes) sujetos en un directorio de entrenamiento y los 4 (744 imágenes) restantes en el de pruebas. Al tener en cuenta el orden de los archivos y la cantidad en cada directorio se procedió a realizar las etiquetas por medio de un documento *Excel*con formato *.xlsx*. En dicho documento se creó una hoja para cada grupo de etiquetas, esto quiere decir, una hoja para etiquetas de entrenamiento y otra para etiquetas de prueba. Por lo que, en cada hoja se puede encontrar una columna con las etiquetas en el orden establecido en la

Figura 2.

### 8.1.1. Etiquetas para clasificación

Para la primera experimentación se consideró clasificar las imágenes en 9 clases diferentes, de tal manera que la orientación de la cabeza respecto de la cámara funcionara de manera similar a una palanca de mandos analógica. A continuación se presenta una visualización de dicha clasificación:

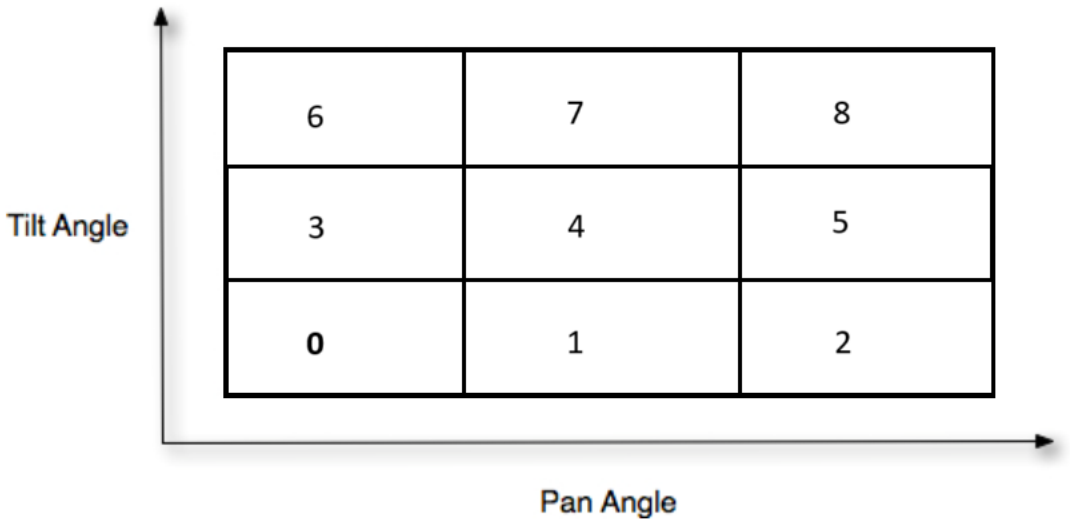


Figura 3: Clasificación de imágenes en 9 etiquetas

Utilizando esta cantidad de clases, se realizaron dos formas de etiquetado en las imágenes, en las cuales solo varía la cantidad de fotografías que fueron etiquetadas dentro de una clase u otra. En las siguientes figuras se podrá observar que la dimensión de cada celda está representada con (n $\times$ m), dónde n corresponde a la cantidad de ángulos panorámicos que abarca y m la cantidad de ángulos de inclinación (según las muestras tomadas del ICPR). Además, la posición de las celdas en las gráficas si corresponde a la numeración de la Figura 2.

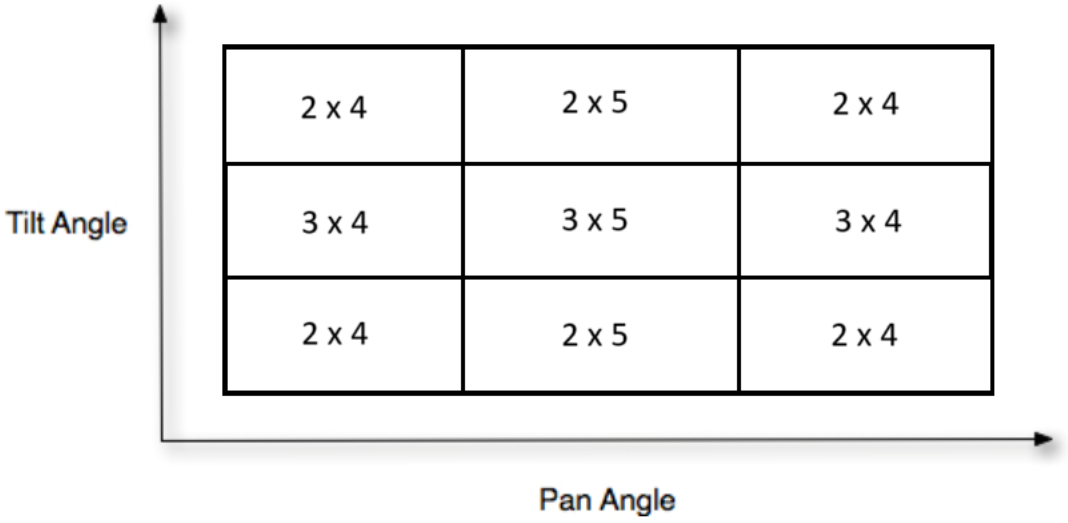


Figura 4: Primera distribución de etiquetas

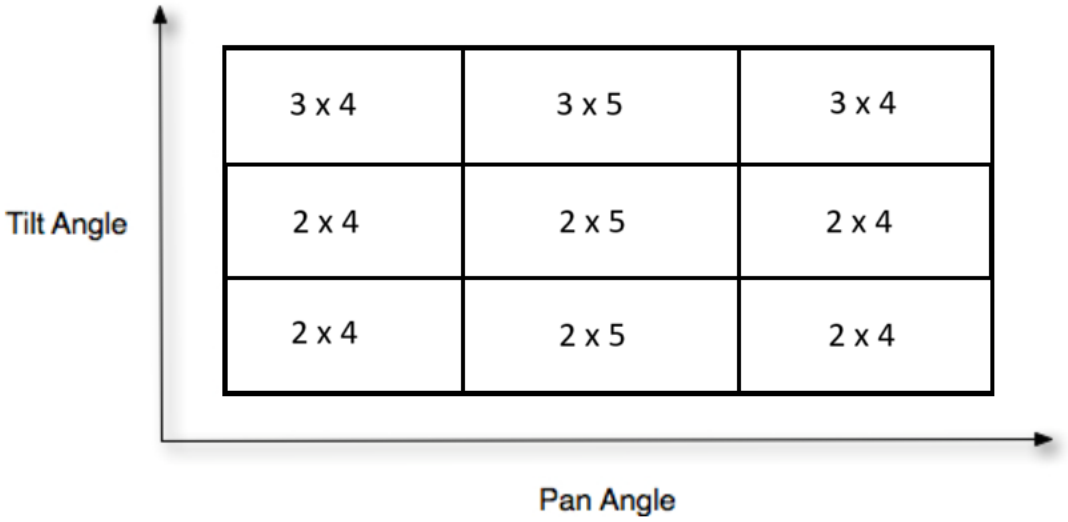


Figura 5: Segunda distribución de etiquetas

Luego de pruebas realizadas con resultados no satisfactorios en la sección de algoritmos para visión por computadora, la siguiente experimentación consistió en reducir la cantidad de clases a 6. En este caso para realizar un movimiento hacia atrás, se requiere un giro completo, en vez del sistema de joystick propuesto anteriormente. La división de clases se muestra a continuación:

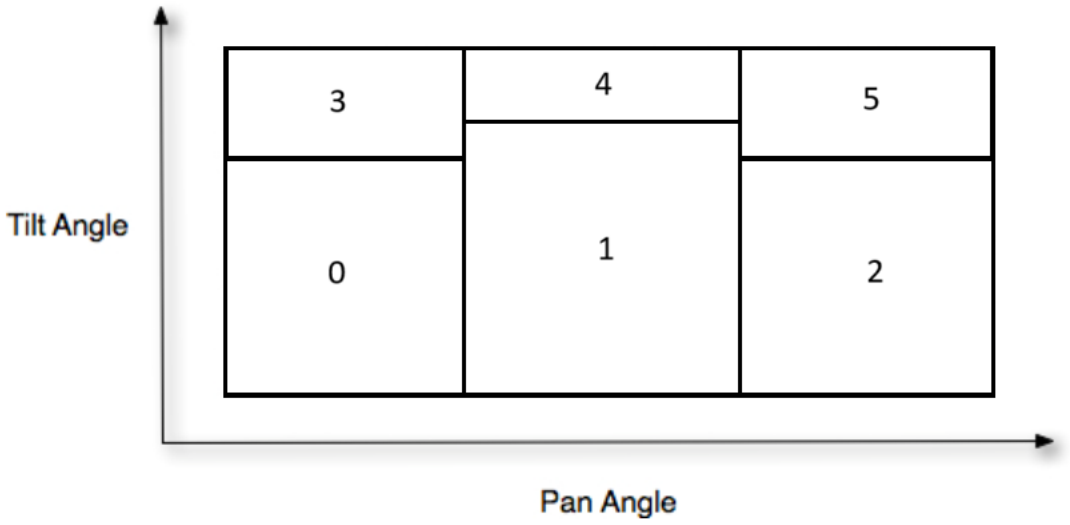


Figura 6: Clasificación de imágenes en 6 etiquetas

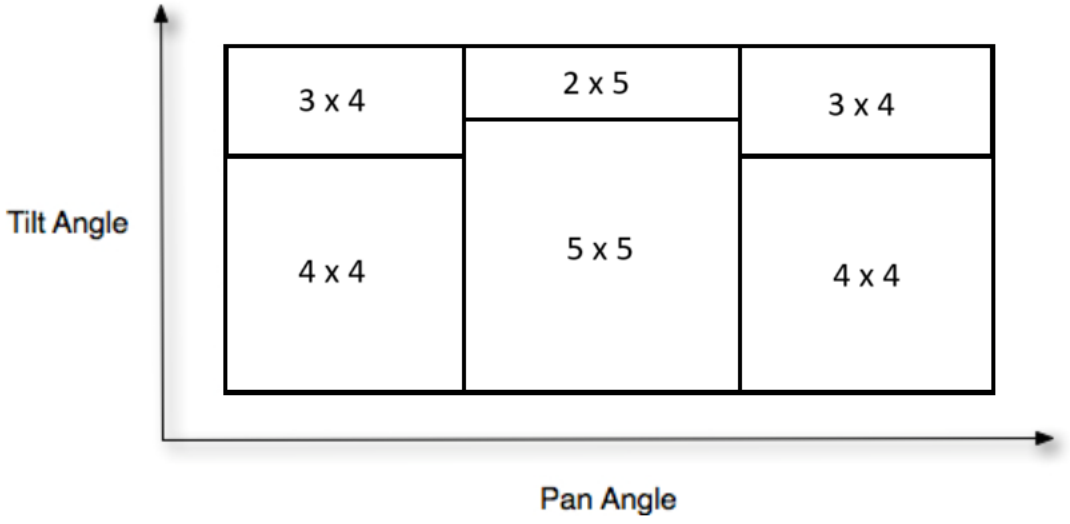


Figura 7: Tercera distribución de etiquetas

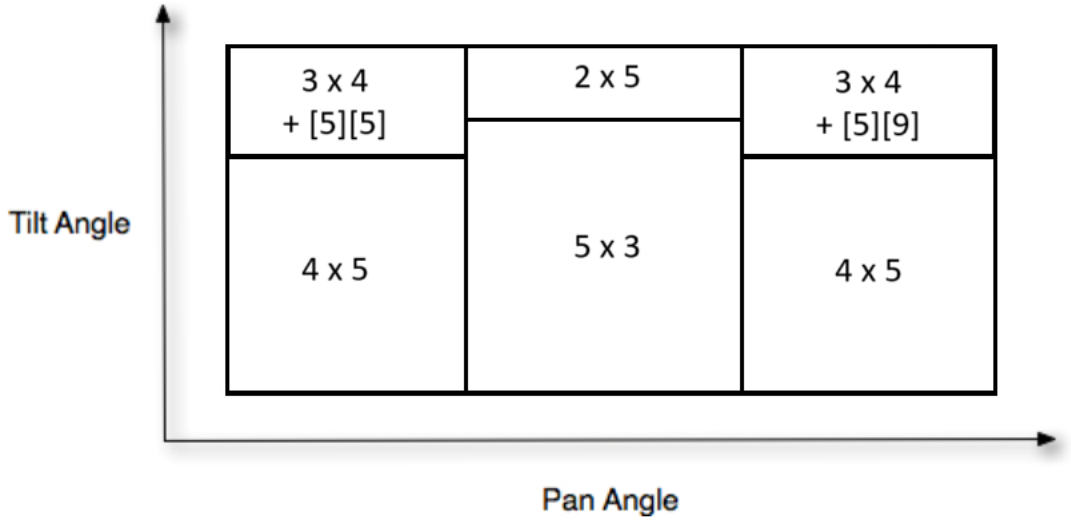


Figura 8: Cuarta distribución de etiquetas (Las posiciones se cuentan desde la esquina inferior izquierda)

## 8.2. Fotografías personales

Dado que los resultados con los modelos de de reconocimiento de orientación iniciales no fueron los esperados en ningún caso, se procedió a realizar 2 series de fotografías personales con la cámara integrada. Cada serie consta de 9 fotografías con posiciones que simulan las 9 clases inicialmente definidas a una distancia de 45 cm aproximadamente. Las series fueron capturadas en un salón de laboratorio del Departamento de Ingeniería Electrónica, Mecatrónica y Biomédica de la Universidad del Valle de Guatemala y en espacios diversos de la casa de Gerardo Andres Fuentes Bámaca (autor). Estas series se agregan al set de entrenamiento, sin embargo, también se requerirá realizar una disminución de clases nuevamente. La nueva cantidad para la primera es de 4, y se tomarán en cuenta la posición de la cabeza en estado neutro, a los lados y hacia arriba o abajo. Luego de obtener el resultado, se realizará una nueva prueba de etiquetas con únicamente 3 clases, al centro y a los lados, de no obtener resultados satisfactorios. Estas series se agregarán a la base de datos utilizada para entrenar los primeros modelos.

### 8.2.1. Primer cambio

Dado que los resultados aún no fueron óptimos en el segundo grupo de modelos, se procedió a utilizar un apoyo visual. En esta caso se utilizaron marcadores en forma de puntos verdes que se colocaron en la frente, mentón, y pómulos de los sujetos de prueba. En esta serie de datos, se obtuvieron datos de 10 sujetos, que voluntariamente se prestaron para la realización de las series de fotografías determinadas en pruebas anteriores con Gerardo Fuentes (autor). Con estas series de fotografías se obtuvieron 968 imágenes que también se clasificaron en 70 % para entrenamiento y 30 % para validación, además de la base de datos

ICPR.

# CAPÍTULO 9

## Algoritmos para visión por computadora

## 9.1. Preprocesado de imágenes

Inicialmente se diseñó un código para extraer fotografías desde un directorio, etiquetas desde un documento con extensión *.xlsx* y luego transformarlos en matrices de datos que para poder ser utilizados en los proceso de *Machine Learning*. Para la realización de este código, se utilizaron las siguientes librerías: *OS*, *CV2 (OpenCV)*, *Numpy* y *Pandas*. Además de esto el proceso para la realización del código fue el siguiente:

### 9.1.1. Transformación de fotografías

En este caso, se utilizó la librería *OS* para crear una lista con nombres en formato cadena de los archivos en un directorio dado. Luego, utilizando la librería *Numpy*, se creó una matriz de ceros para guardar todas las fotografías convertidas en formato de matriz. Para realizar la conversión, previamente se realizó una prueba para obtener las dimensiones mínimas con las que puede trabajar la cámara integrada con la librería *CV2*, cuyo tamaño es $320 \times 180$ píxeles. Después, dentro de un bucle *for* se utilizó la librería *CV2* obtener la matriz de pixeles de cada fotografía, ajustar el tamaño a las dimensiones encontradas y luego transformar el canal de color por defecto BGR *(Blue, Green, Red)* a RGB *(Red, Green, Blue)*.

### 9.1.2. Transformación de etiquetas

Para transformar las etiquetas en matrices de datos, únicamente se requirió la librería *Pandas*, ya que nos permite utilizar una función específica para leer documentos con formato *.xlsx*, así como acceder a las hojas y columnas del documento. Al guardar el resultado de esta función el siguiente paso es convertirlo a un formato de matriz con funciones de *Numpy*.

### 9.1.3. Exportación de datos finales

Antes de realizar la exportación de datos, es importante aplicar una transformación de tipo flotante a entero, en las matrices, y de esta manera se ahorra espacio en el disco duro. Ahora bien, la librería *Numpy* cuenta con una función para exportar matrices de datos de manera individual (una variable a la vez), o bien agruparlas en un solo archivo. En este caso, dado que se necesitan realizar diferentes pruebas, se estará exportando de manera individual, en cambio para los resultados finales, se utilizará la función de guardar de manera agrupada.

### 9.1.4. Filtrado de imágenes (Modelo No. 8)

Luego de las pruebas con los primeros 7 modelos de reconocimiento facial, dónde únicamente se utilizaban los datos de las fotografías a un tamaño de $320 \times 180$ píxeles, con canal RGB, para alimentar el proceso, se decidió utilizar la librería *OpenCV* para aplicar filtros de bordes a las fotografías, de tal manera que el modelo de aprendizaje no tome en cuenta los colores sino únicamente los bordes de los rostros. Este proceso se aplicó en la sección del código par transformar las fotografías, y en la sección que recopila la imagen en tiempo real.

## 9.2. Entrenamiento de modelos de *machine learning*

Para las primeras pruebas se utilizó el framework *TensorFlow y Keras* para poder realizar los diferentes modelos de *Machine Learning* para reconocimiento de orientación de cabeza, utilizando las matrices de datos obtenidas a partir de las bases de datos recopiladas. Para este código se utilizaron las librerías *TensorFlow*, *Matplotlib*, *Numpy* y *Seaborn*. El código fue realizado de la siguiente manera:

### 9.2.1. Extracción de datos previamente procesados

Luego de exportar los archivos transformados de la base de datos, se generan archivos con extensión *.npy* para los archivos de una variable y *.npz* para los de múltliple variables. Dichos archivos pueden ser importados utilizando también funciones de la librería *Numpy*, para guardarlos en variables utilizables dentro del código, optimizando así el tiempo de transformación de datos.

### 9.2.2. Modelo de reconocimiento de orientación de cabeza

El primer pasó fue normalizar los valores de las matrices, dado que son imágenes RGB, cada píxel de estas está representado por 3 valores que varían entre 0 y 255, y se necesitan valores entre 0 y 1, por lo que únicamente se tuvo que dividir tanto la variable con datos de entrenamiento y prueba entre el valor 255.

**Primer grupo de modelos de aprendizaje**

Inicialmente, dado que se trata de un modelo que trabaja con matrices derivadas de fotografías, se utilizó una CNN (Red Convolucional Neurnal), la cual permite detectar patrones en las imágenes. Utilizando *Tensorflow* y *Keras*, se definieron 7 capas para la generación del modelo. En este caso se utilizó en primera instancia una capa convolucional en 2D, con 10 kernels de tamaño $3 \times 3$, función de activación *ReLU*, y por ser la primera capa se agregan las dimensiones de entrada (fotografía). Luego, se aplicó una capa con la técnica de *Max Pooling* con tamaño $2 \times 2$. Después, se aplica una capa de aplanado previa necesaria para poder utilizar las siguientes capas densas. Esta vez se utilizaron 3 capas densas, con 75, 150 y 75 nodos respectivamente con función de activación *ReLU*. Por último, se aplica una capa densa con una cantidad de nodos definida por el número de clases a utilizar y función de activación *Softmax*. Con esta especificación de hiperparámetros, se procede a unir las capas en un modelo secuencial, para luego compilarlo con un optimizador *Adam*, para la pérdida se utilizó *Sparse categorical crossentropy*, dado que las etiquetas fueron definidas como enteros y no en el formato *One-Hot*. Para finalizar con los hiperparámetros, en la opción métricas se escoge la opción *Accuracy* o precisión. Luego de entrenado el modelo, se procede a guardarlo con la función de guardar en *Keras*, en el cual el argumento es el directorio donde se desea guardar el archivo con extensión *.keras* y poder utilizarlo en otros códigos.

Al tener configurado el modelo, se procede al entrenarlo, guardarlo y exportar las gráficas descriptivas respectivas. En esta etapa se cuenta con 4 modelos que varían dado que se utilizó cada uno de los diferentes grupos de etiquetas. A continuación se muestran las gráficas de pérdida y exactitud para cada modelo, utilizando el grupo de entrenamiento actualizado con las fotografías de Gerardo Fuentes (autor), así como las matrices de confusión utilizando la librería *Seaborn*:

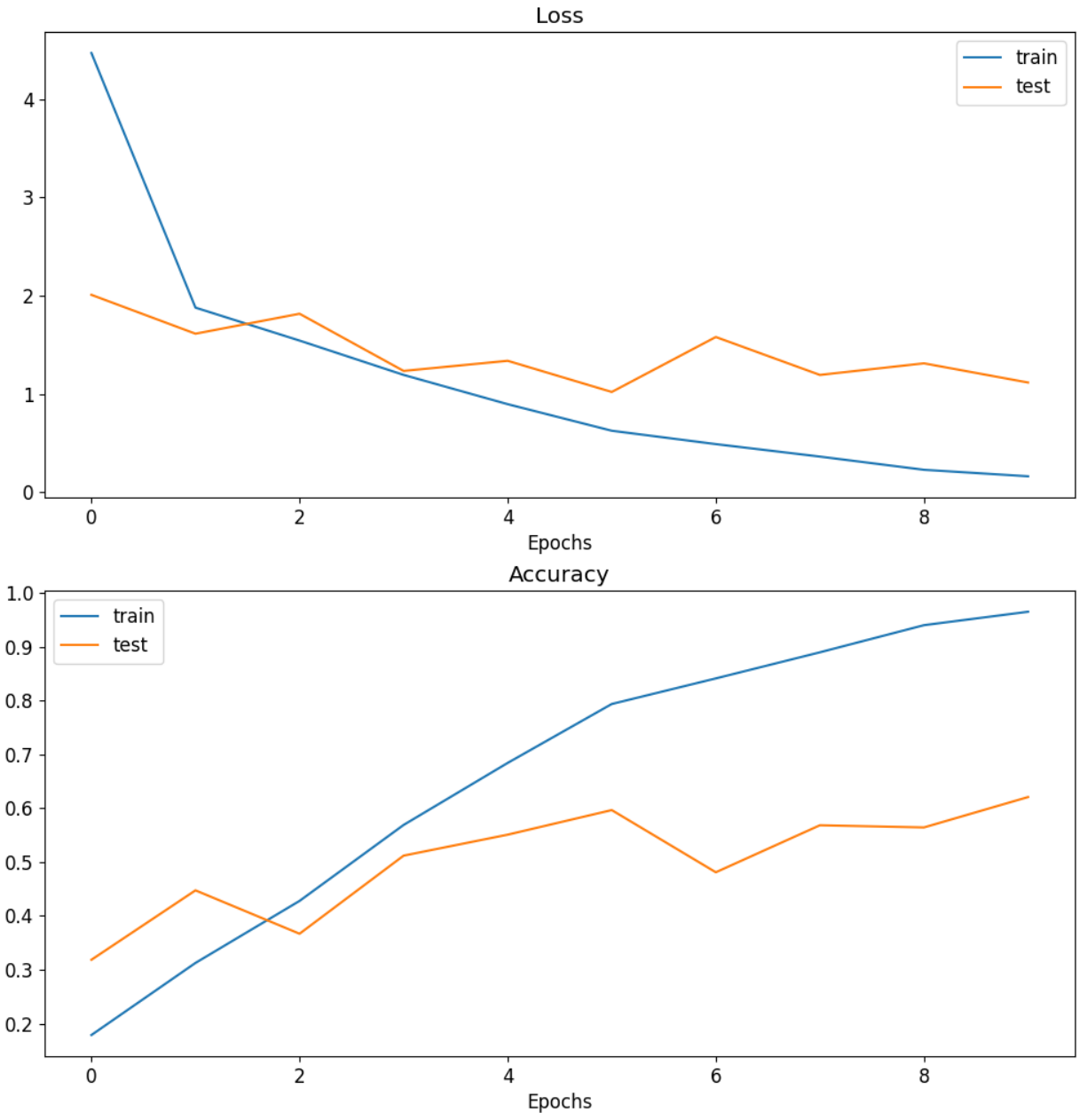


Figura 9: Pérdida y exactitud del primer modelo con 9 clases

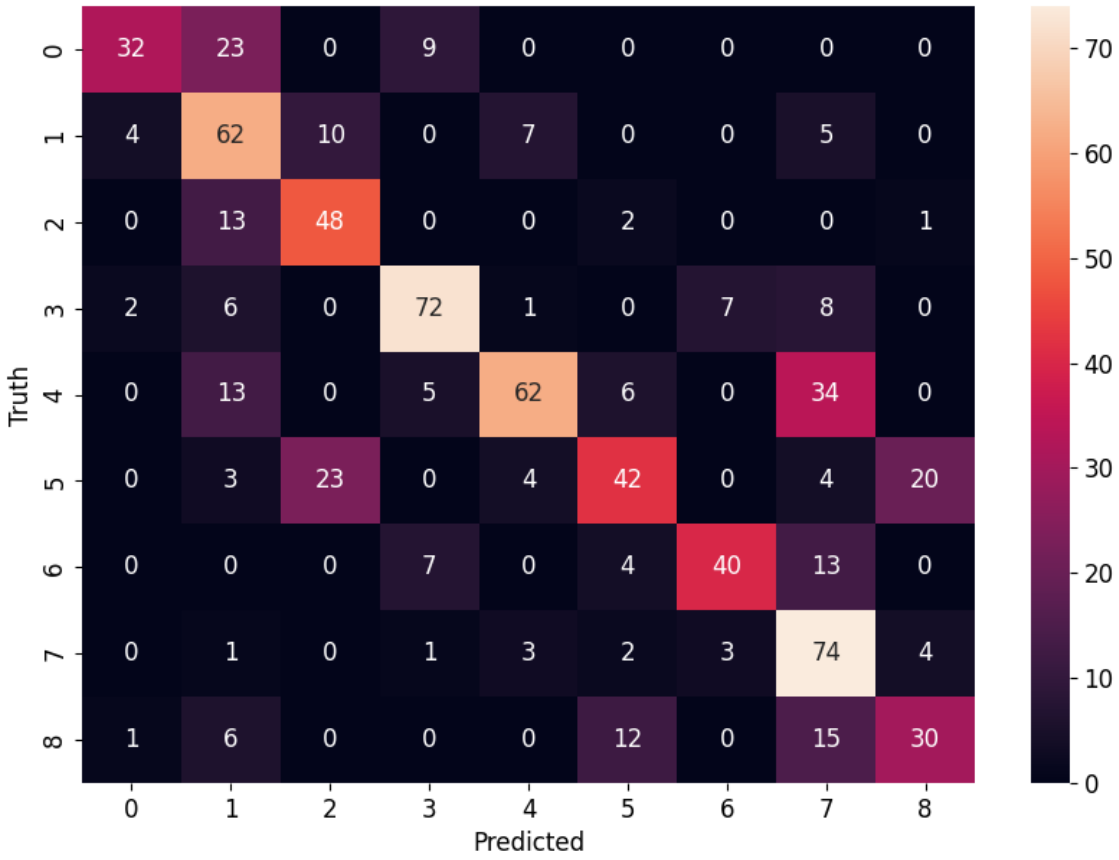


Figura 10: Matriz de confusión del primer modelo con 9 clases

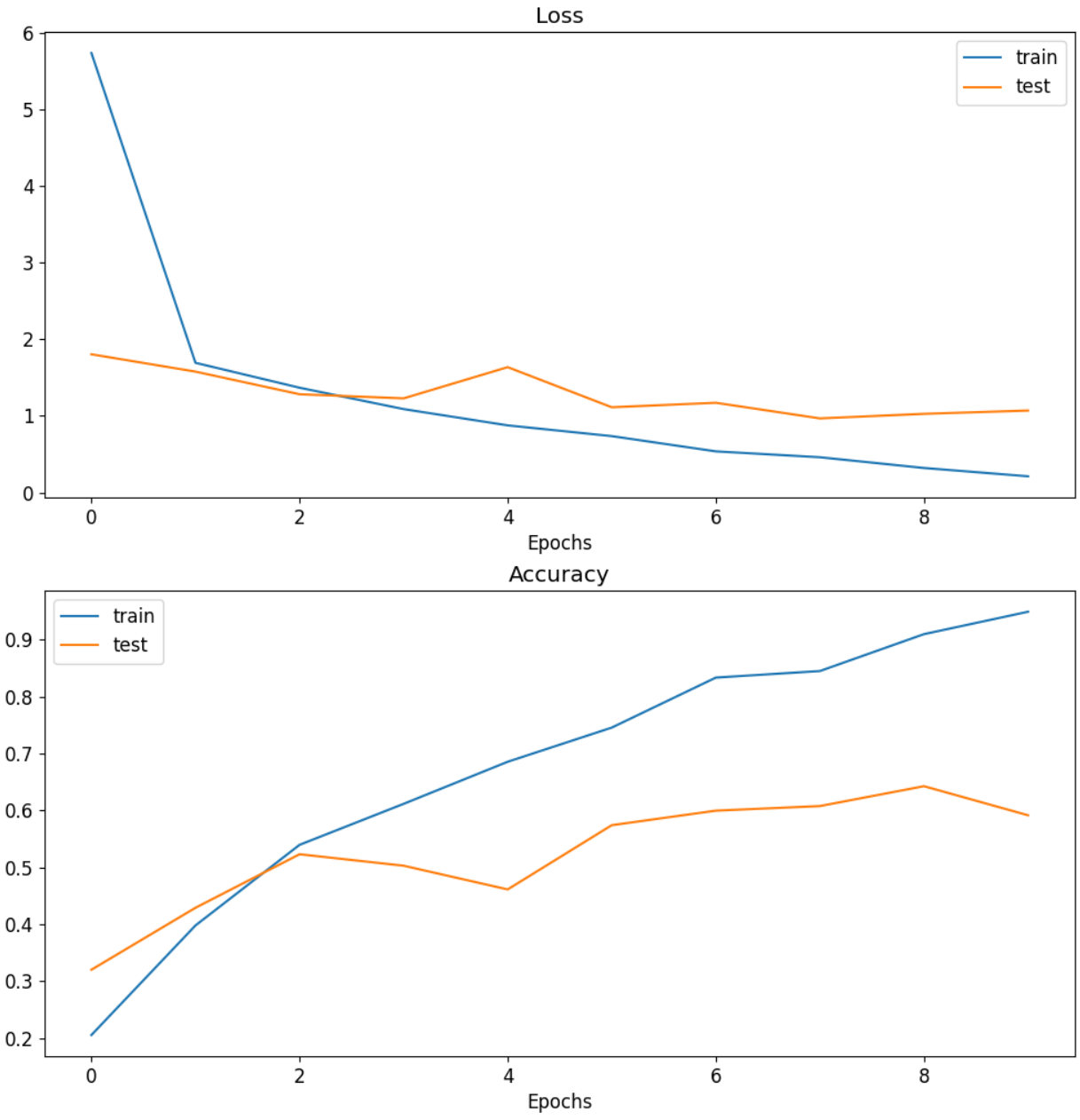


Figura 11: Pérdida y exactitud del segundo modelo con 9 clases

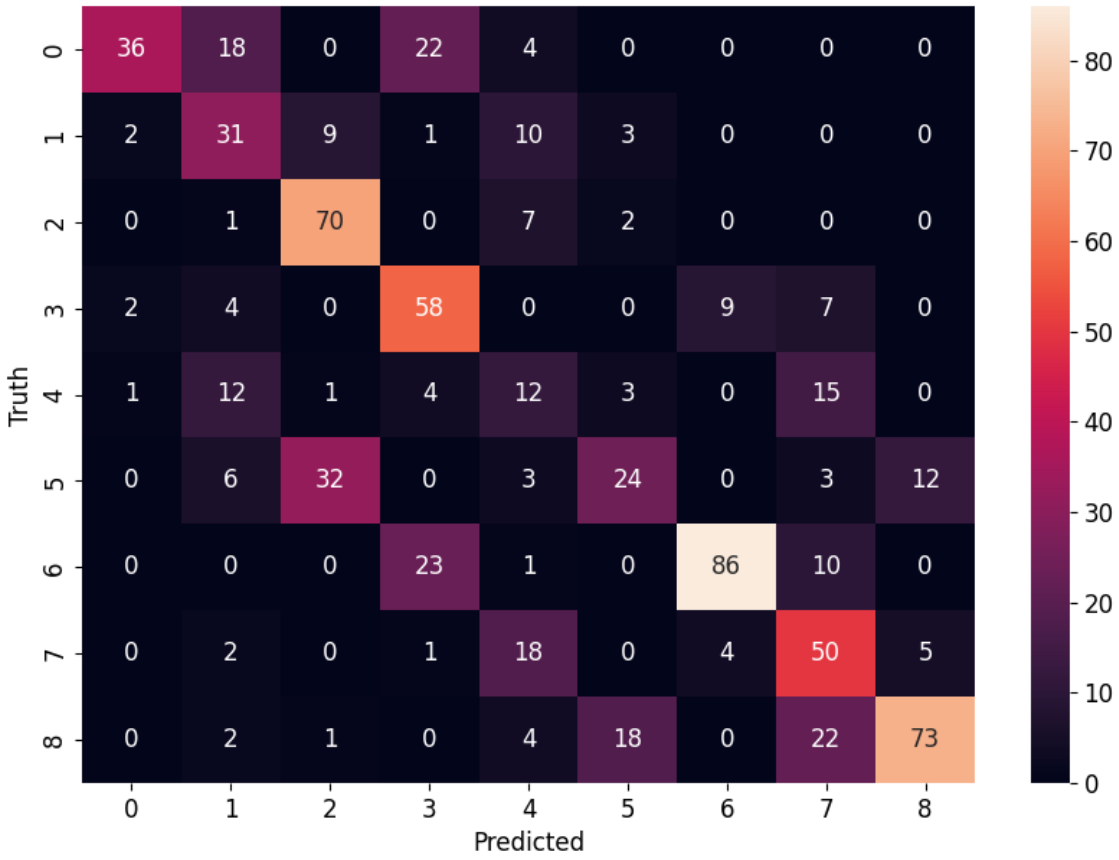


Figura 12: Matriz de confusión del segundo modelo con 9 clases

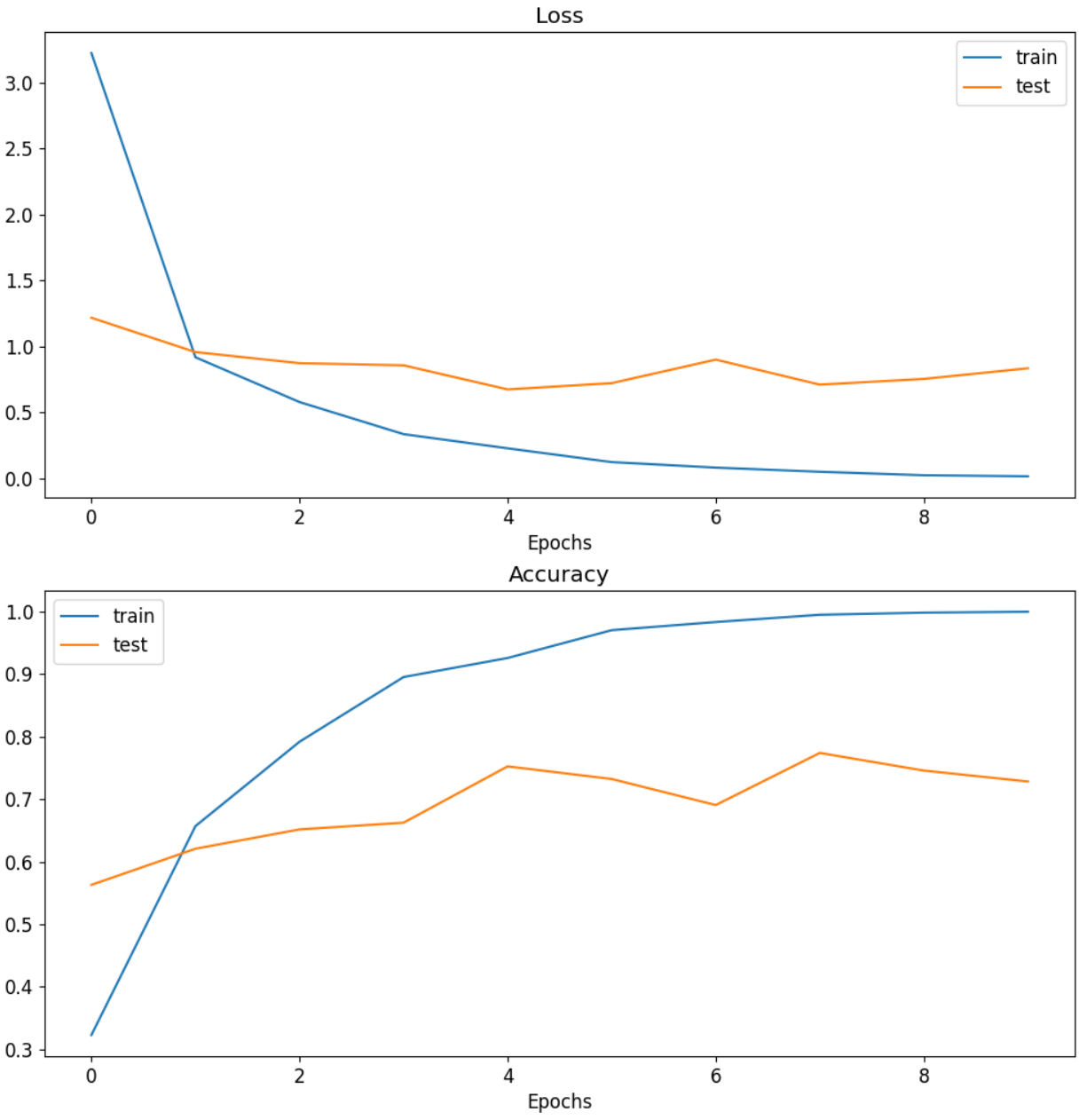


Figura 13: Pérdida y exactitud del primer modelo con 6 clases

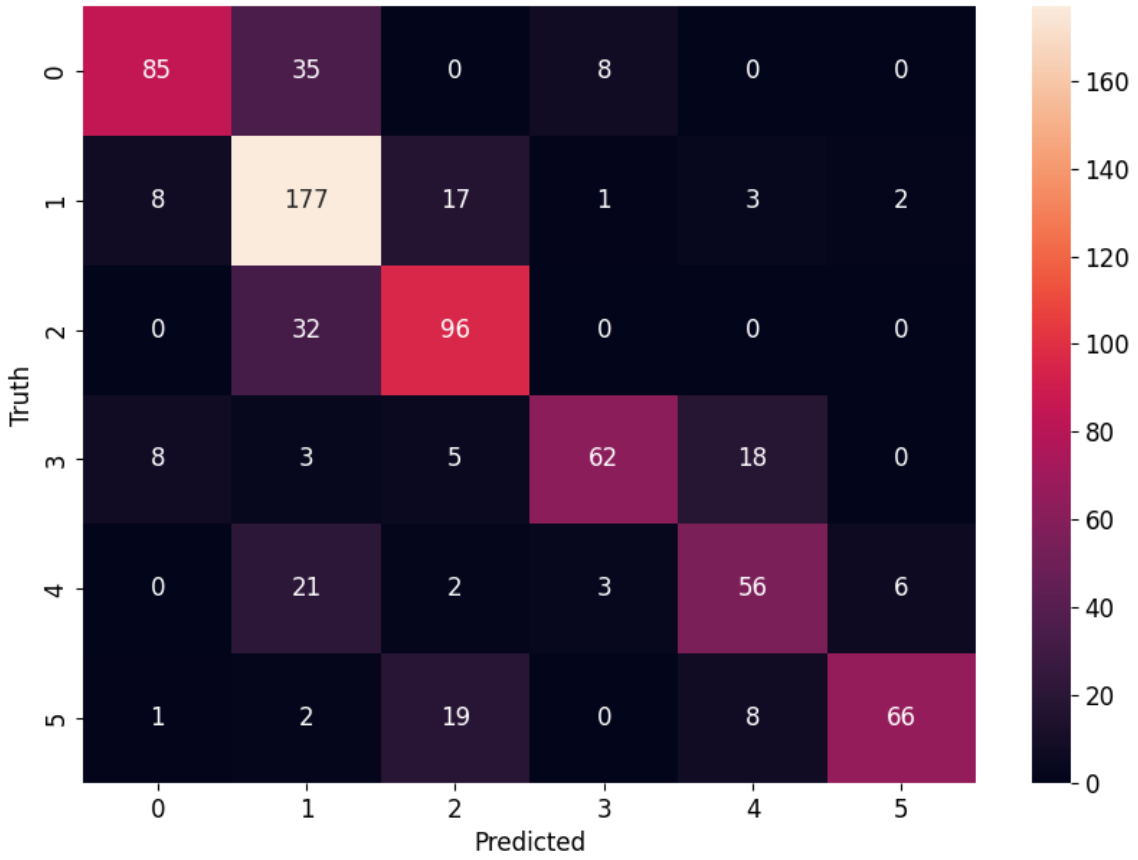


Figura 14: Matriz de confusión del primer modelo con 6 clases

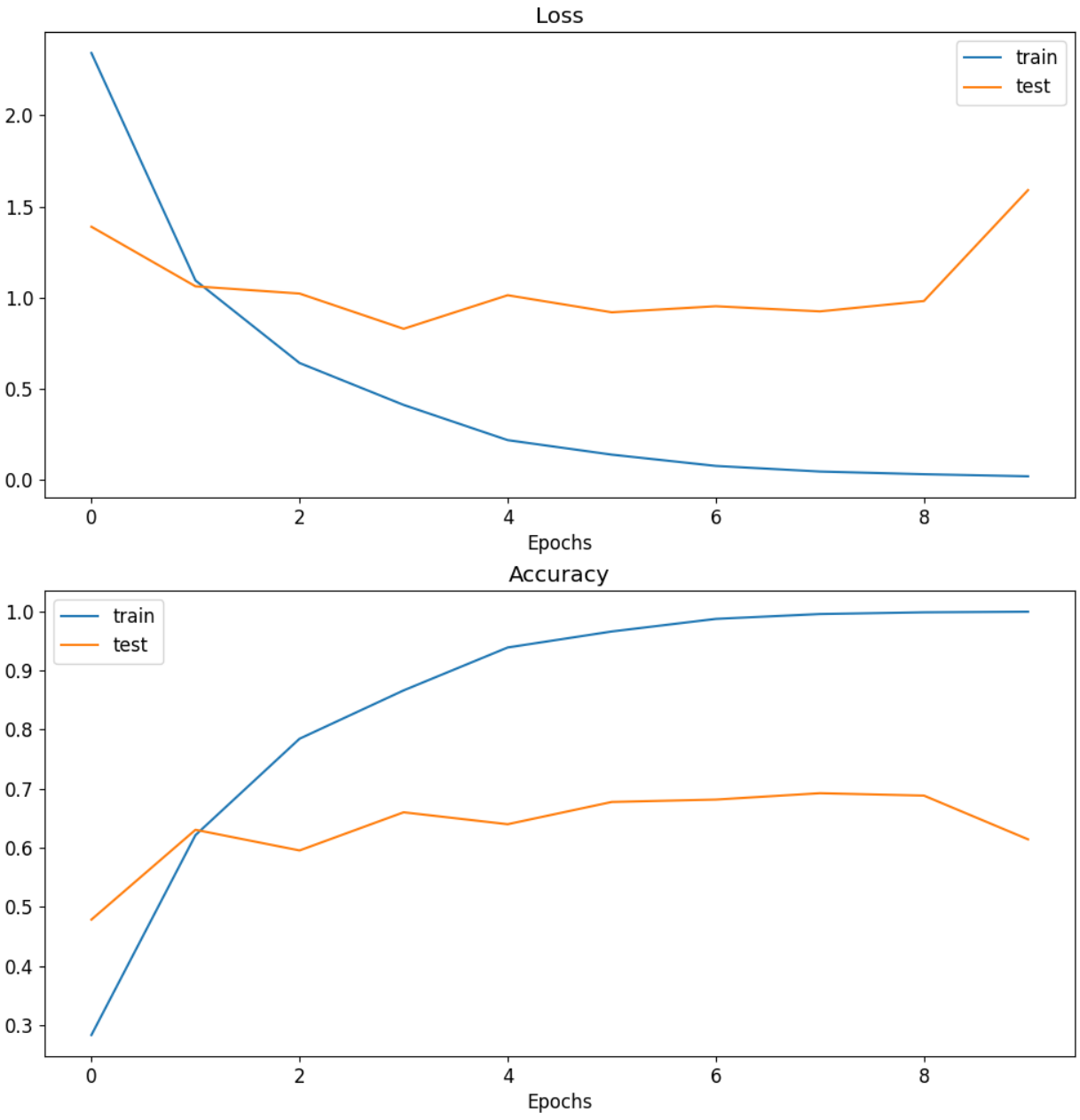


Figura 15: Pérdida y exactitud del segundo modelo con 6 clases

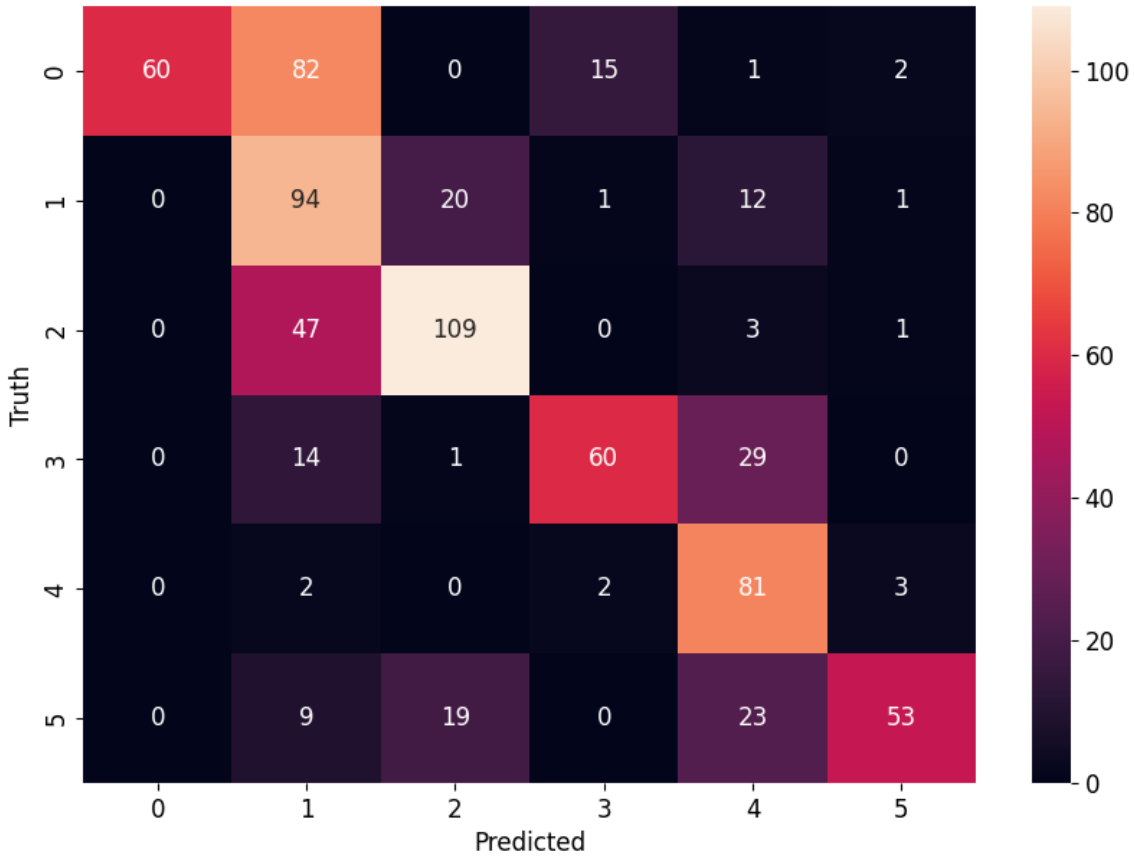


Figura 16: Matriz de confusión del segundo modelo con 6 clases

En las figuras 9 a las 16 se puede observar que se produce un sobreajuste, tanto en pérdida como en exactitud en todos los modelos entrenados. Los valores en pérdida de validación no disminuyen más allá del valor 1.0 y la exactitud en validación no supera el 70 %. Además, en las matrices de confusión, de todos los modelos, se observa que la predicción falla al detectar la posición vertical. Esto sucede dado que las fotografías de la base de datos ICPR, tienen el mismo fondo en todos su casos. Además, la mayoría de los sujetos de prueba son de sexo masculino, caucásicos y con un rango de edad similar. Además de ser únicamente 11 personas diferentes. Con dichos resultados, se pueden tomar diferentes decisiones. Las acciones a considerar son:

- Conseguir más fotografías y con más diversidad de sujetos y fondos.
- Reducir nuevamente la cantidad de clases a 3 o 4.
- Realizar un sobreajuste enfocado en el rostro del usuario.

**Segundo grupo de modelos de aprendizaje**

En este caso se utilizó la misma configuración para el modelo de aprendizaje utilizado en los primeros 4 modelos, por lo que únicamente se varió la cantidad de etiquetas utilizadas. A continuación se muestran las gráficas descriptivas para el modelo de 4 clases:

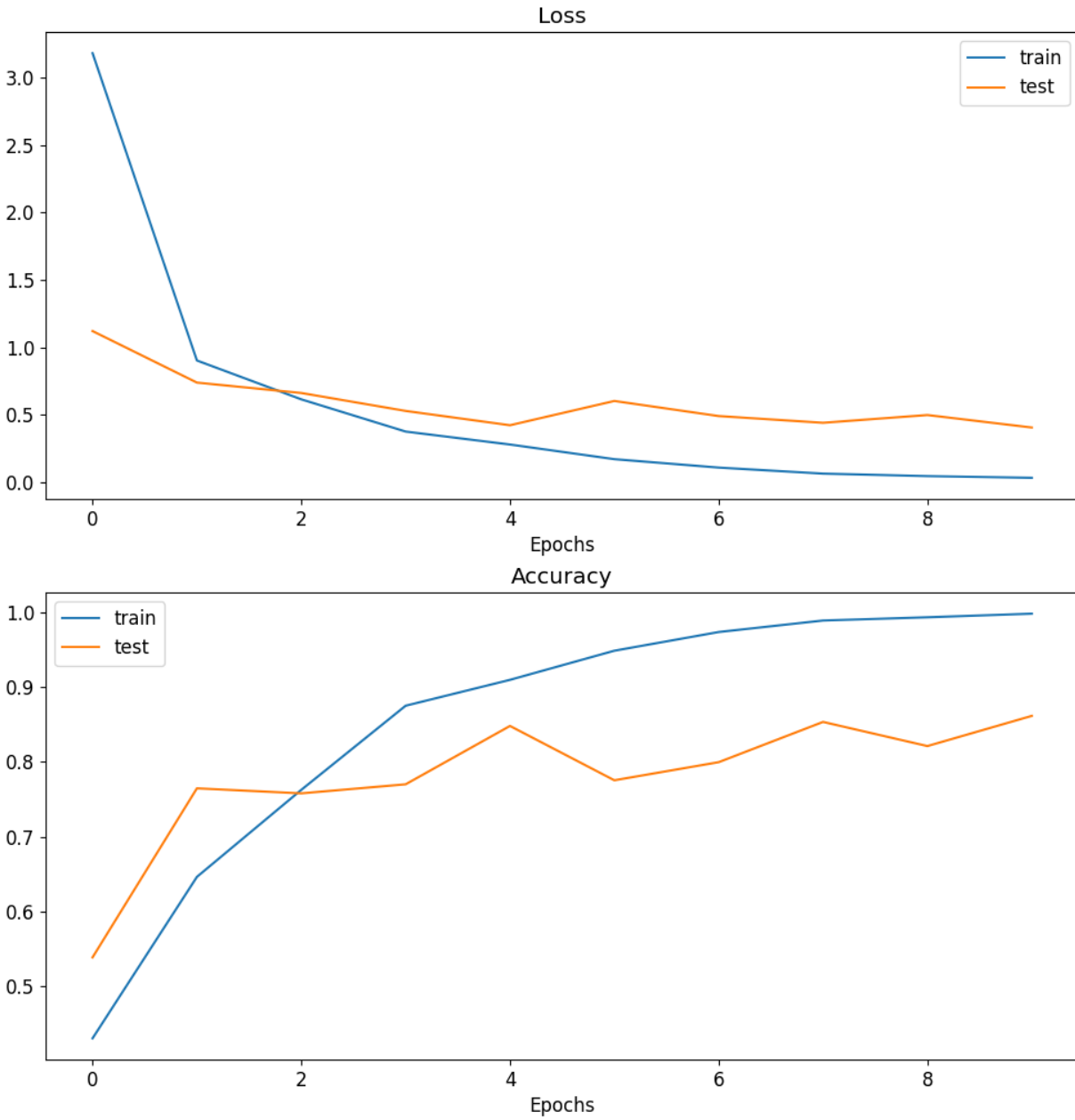


Figura 17: Pérdida y exactitud del modelo con 4 clases

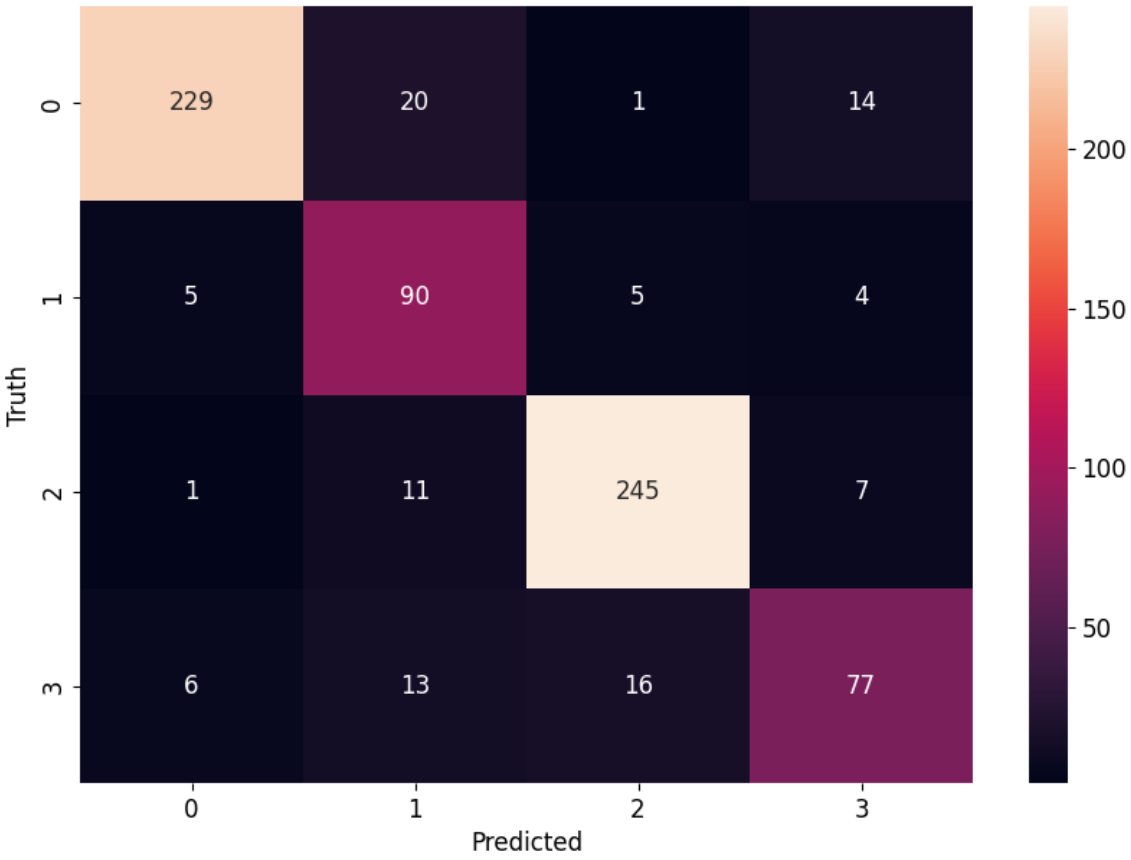


Figura 18: Matriz de confusión del modelo con 4 clases

Al observar la figura 17, se resalta una disminución en la pérdida en validación, que anteriormente convergía aproximadamente en los valores de 1.0 y ahora se acerca a 0.5. En la gráfica de exactitud, por primera vez se logró superar el 80 % en validación. Además, el comportamiento de la exactitud se aprecia de manera visible en la matriz de confusión en la figura 18.

A continuación se muestran las gráficas descriptivas para el modelo de 3 clases:

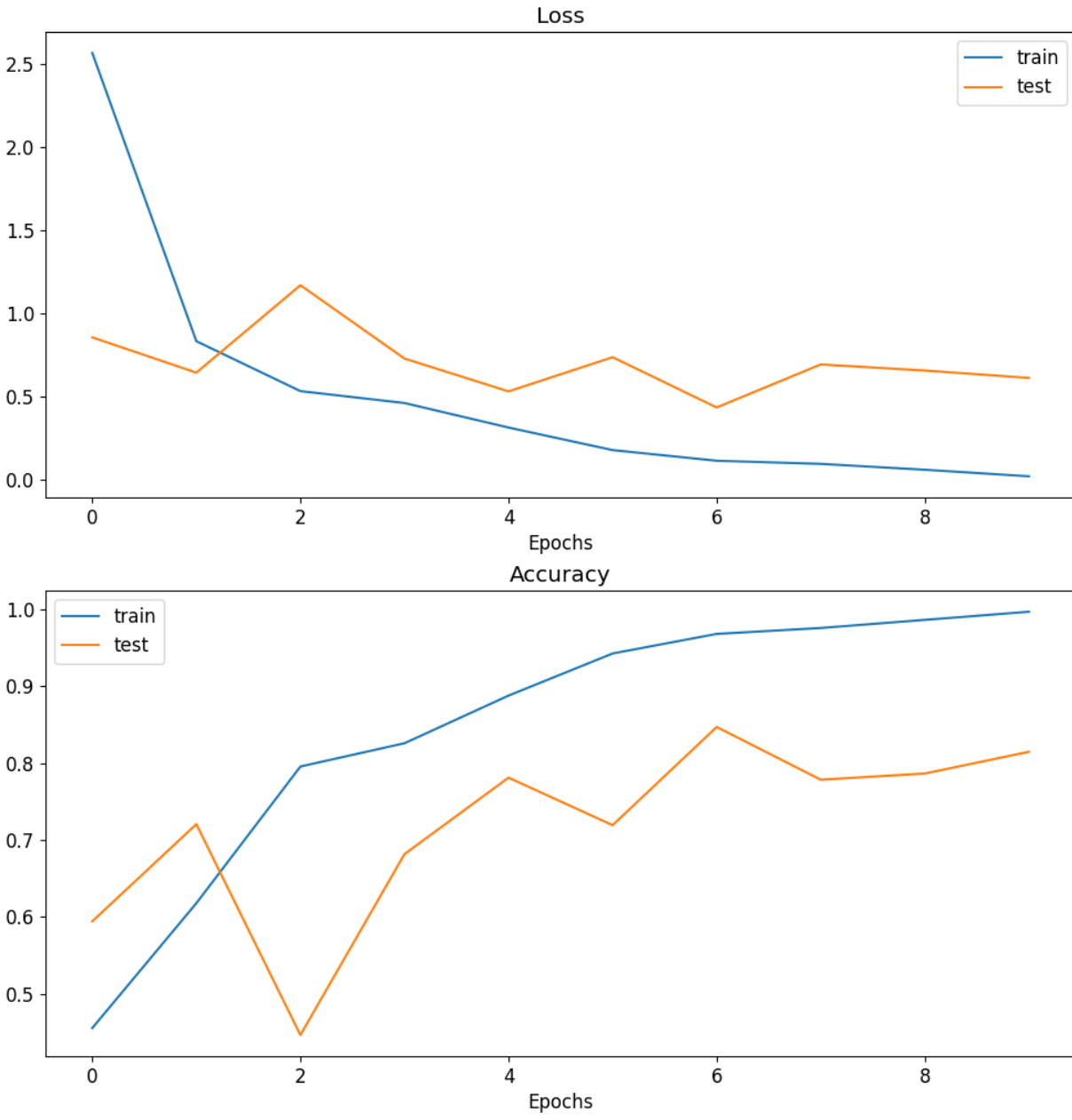


Figura 19: Pérdida y exactitud del modelo con 4 clases

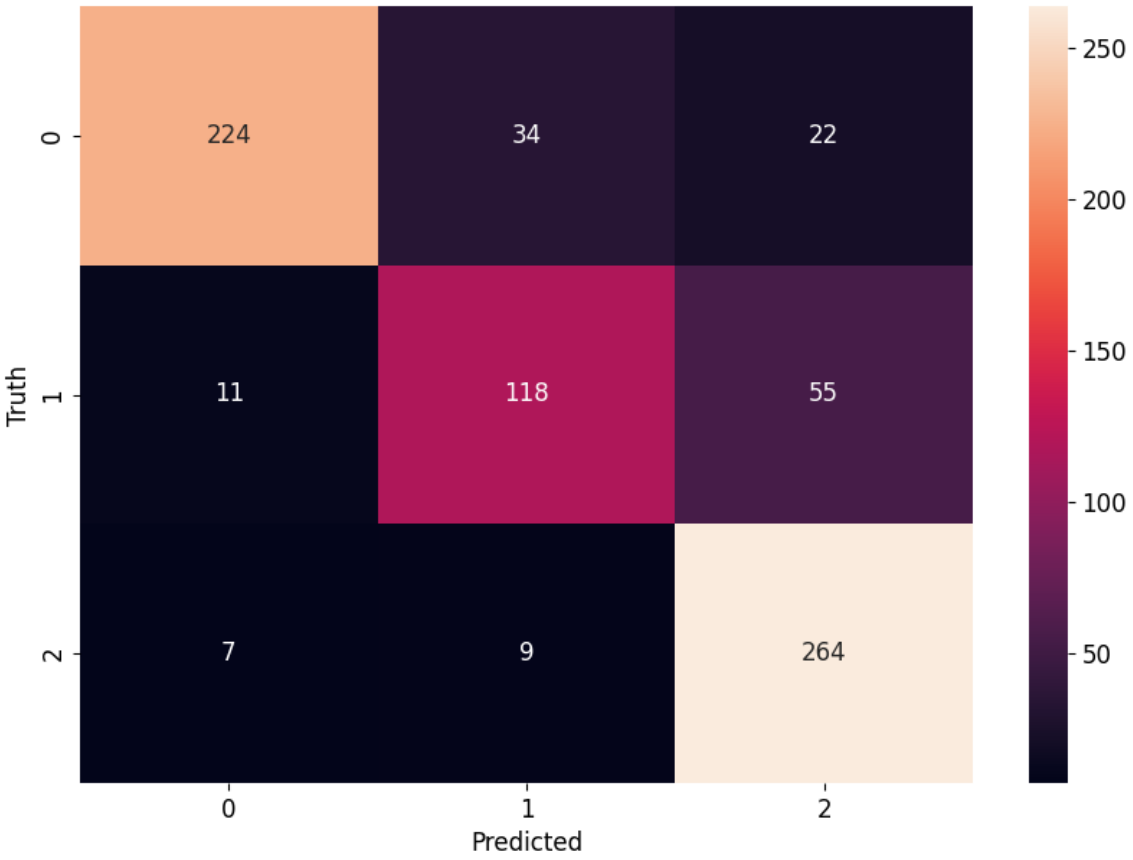


Figura 20: Matriz de confusión del modelo con 4 clases

En la figura 19 se puede observar que la pérdida fluctúa entre los valores 0.5 y 0.1, lo cual es muy similar al modelo de 4 etiquetas. En la gráficas de exactitud, se observa que los datos predecidos nuevamente generan una exactitud levemente superior al 80 %. Por otro lado, en la matriz de confusión en la figura 20, se observa que las magnitudes de los aciertos son similares entre sí respecto de las clases. En general el modelo de 3 clases tiene una buena base teórica para funcionar con el usuario Gerardo Fuentes, sin embargo utilizar 4 clases resulta más útil a nivel de comandos.

### Tercer grupo de modelos de aprendizaje

En este grupo de modelos, se utiliza la misma configuración para la red neuronal que en los modelos anteriores. Sin embargo, en este caso se incrementó la cantidad de datos para entrenamiento, se utilizan 4 clases en todos los modelos, el octavo modelo utiliza filtros y el noveno fue cargado con imágenes de sujetos de prueba que utilizan marcadores faciales, los cuales se explican en la sección de bases de datos y fotografías personales. A continuación

se muestran los resultados de los modelos:

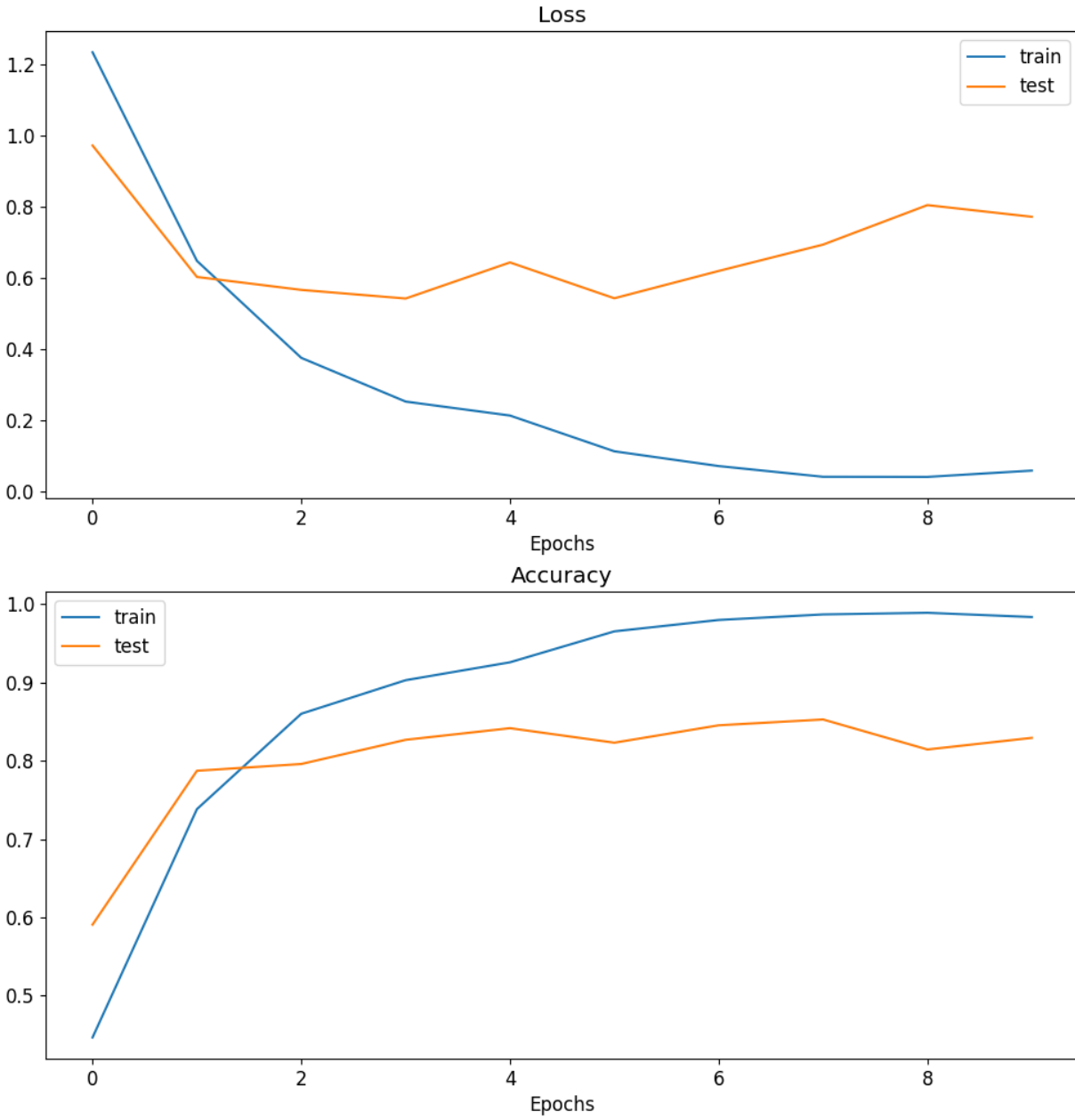


Figura 21: Pérdida y exactitud del modelo con 4 clases y aumento de imágenes de entrenamiento

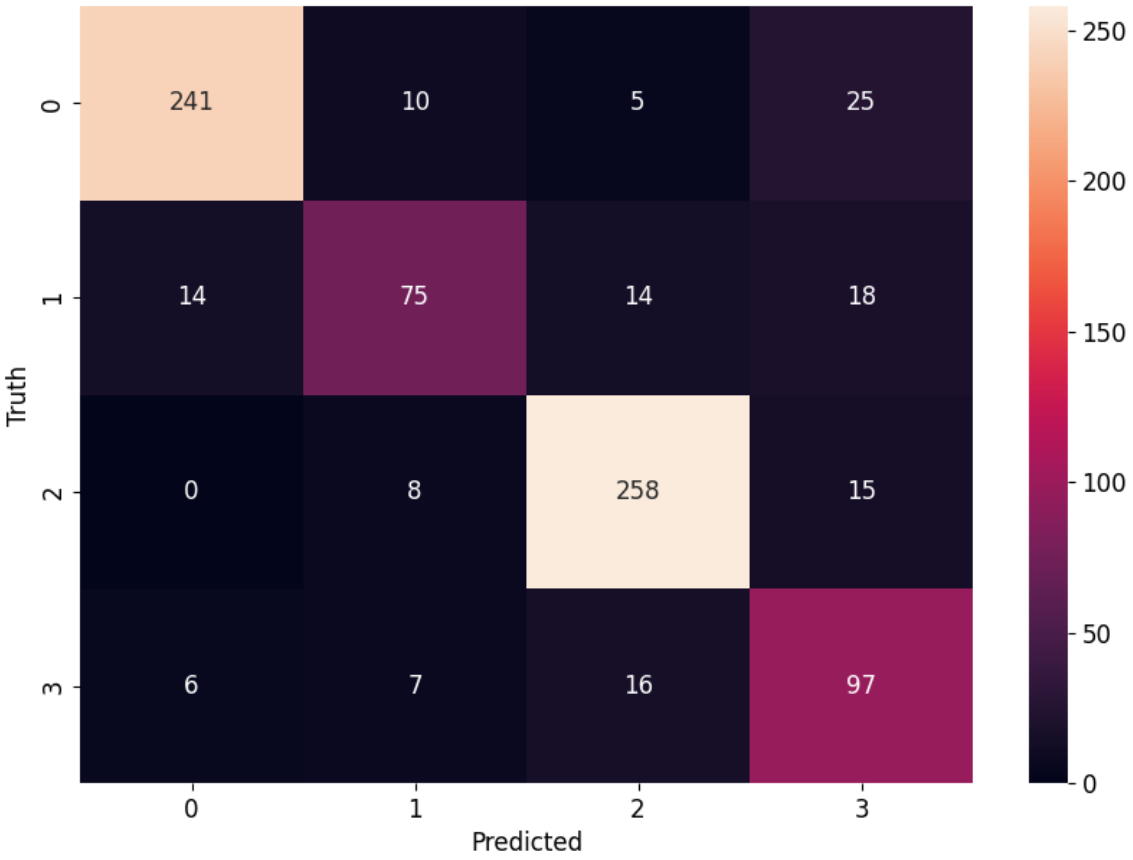


Figura 22: Matriz de confusión del modelo con 4 clases y aumento de imágenes de entrenamiento

En la figura 21 se puede observar que los valores de pérdida por primera vez lograron valores menores a 1.0, y la exactitud logra superar el 80 % de aciertos. Además la matriz de confusión de la figura 22 muestra una diagonal de aciertos mucho más clara.

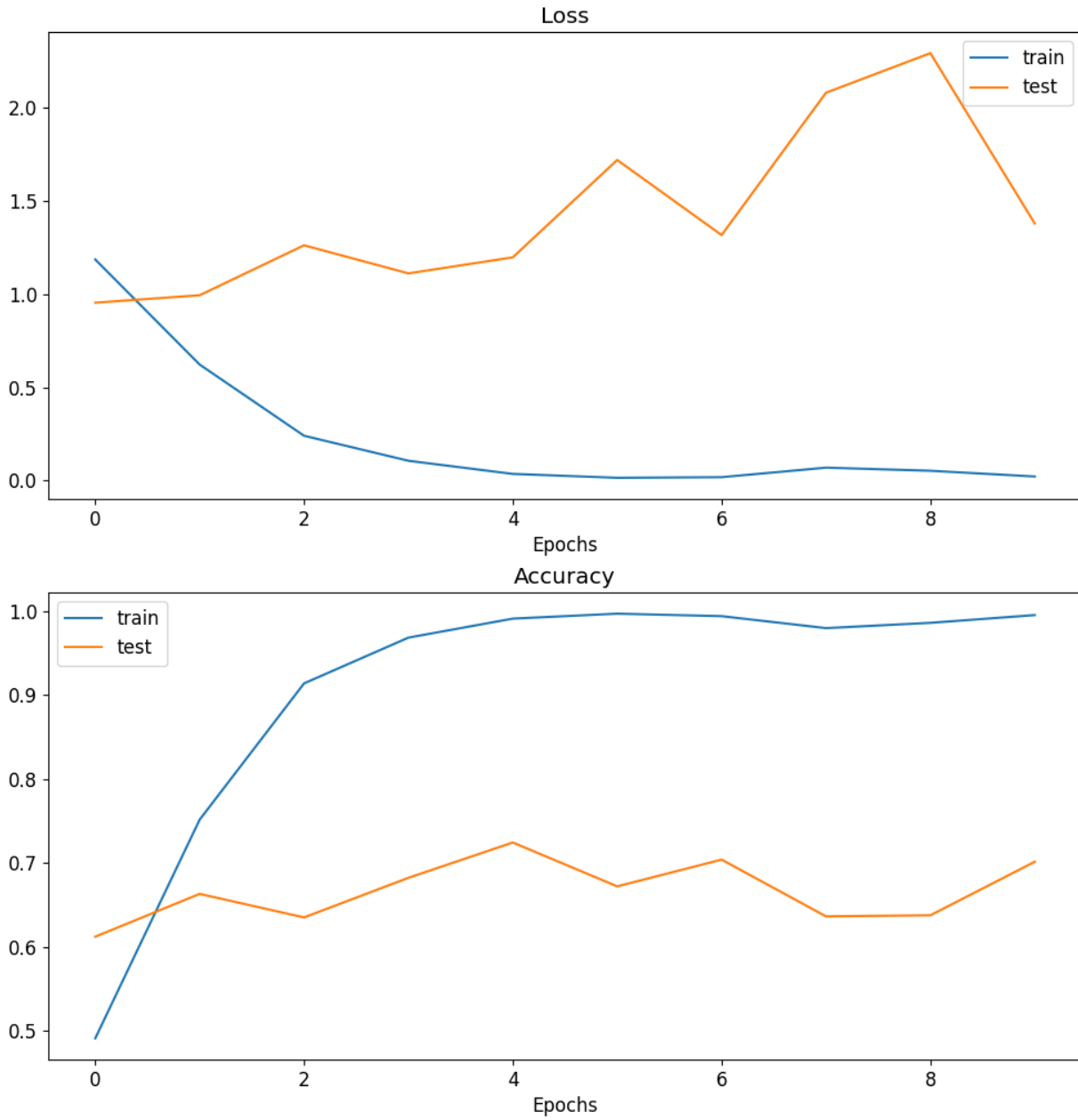


Figura 23: Pérdida y exactitud del modelo con 4 clases con filtro de bordes

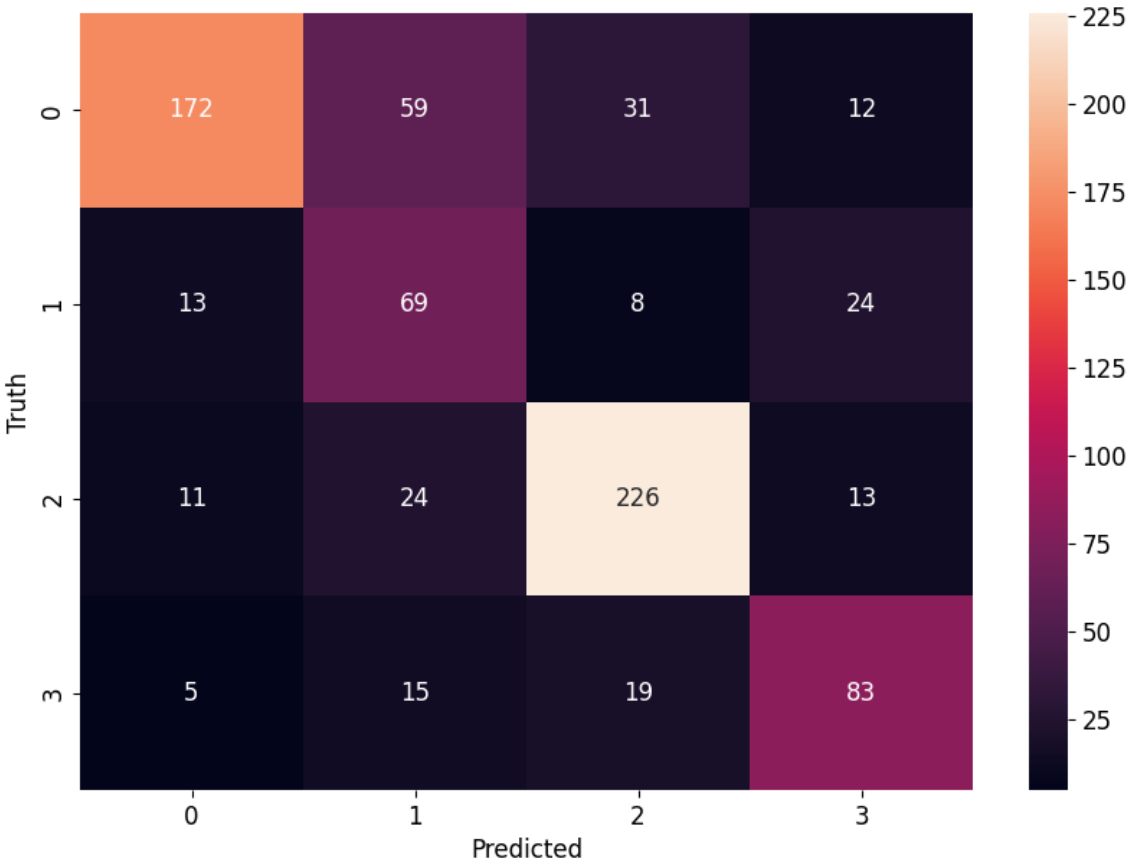


Figura 24: Matriz de confusión del modelo con 4 clases con filtro de bordes

En este caso se puede observar el octavo modelo, el cual incluye un filtro de bordes para las fotografías. En la figura 23, se muestra un claro paso atrás en la disminución de valores de pérdida, superando nuevamente el valor de 1.0, mientras que para la exactitud de aciertos, el porcentaje queda aún menor a 70 %. Este comportamiento erróneo es visible en la matriz de confusión de la figura 24.

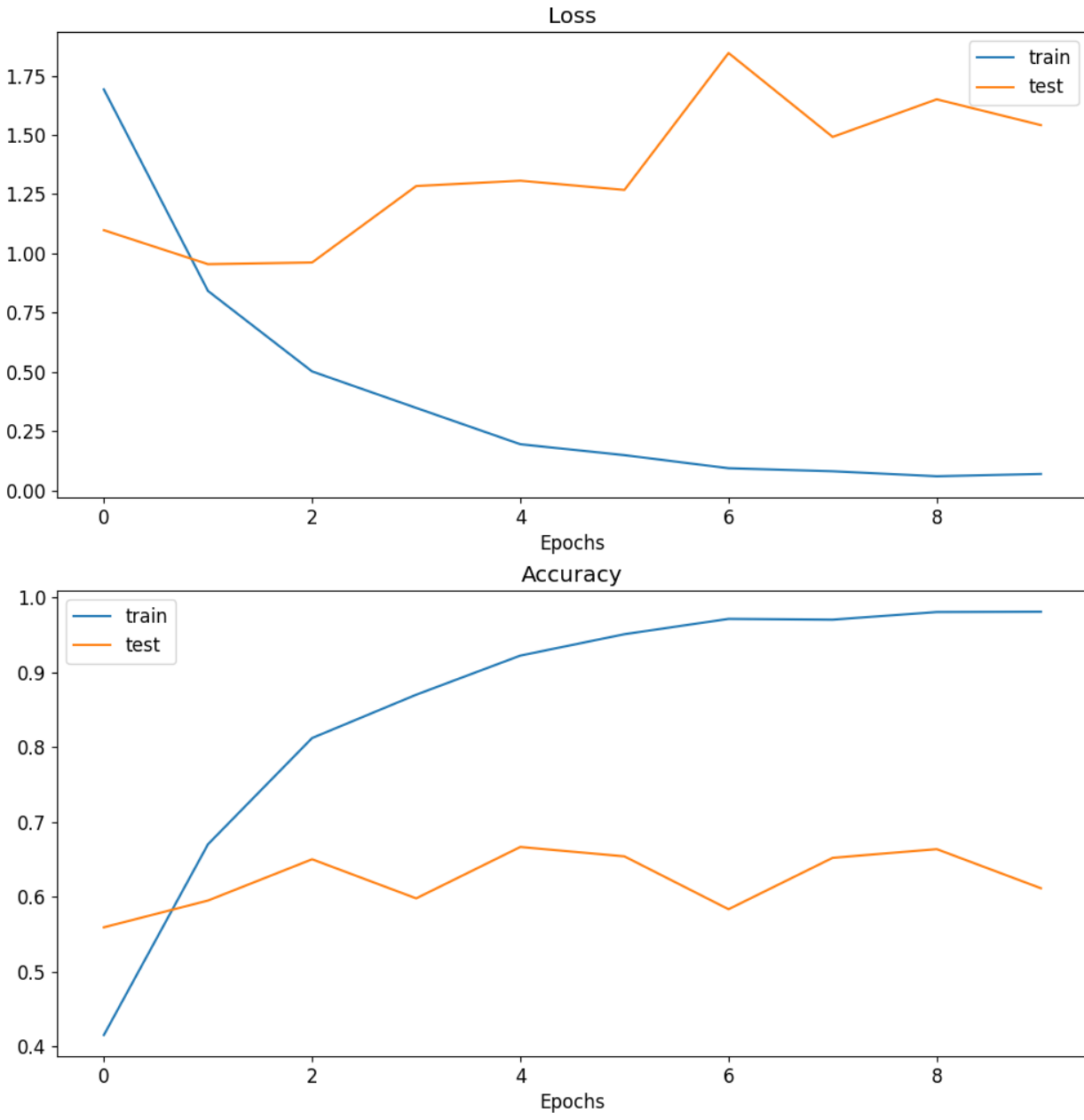


Figura 25: Pérdida y exactitud del modelo con 4 clases y marcadores faciales

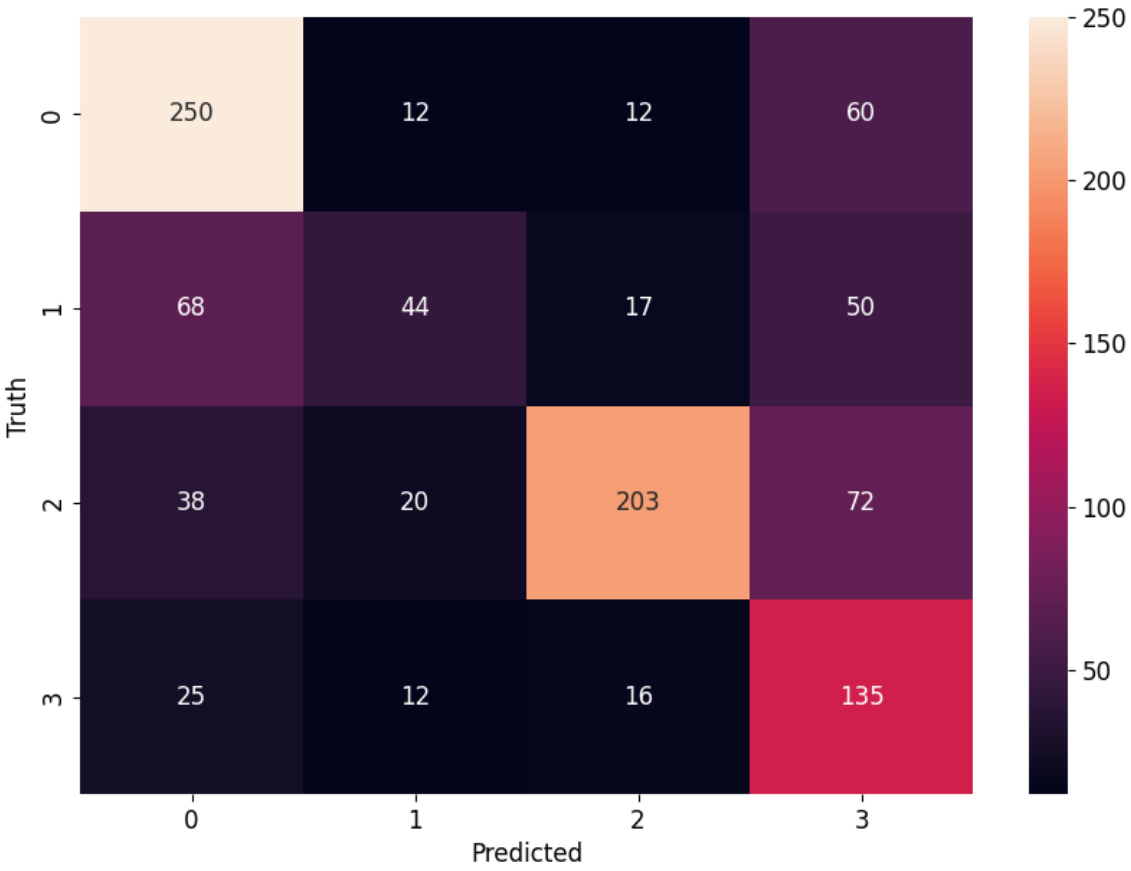


Figura 26: Matriz de confusión del modelo con 4 clases y marcadores faciales

Para este noveno modelo, en la figura 25 el resultado de pérdida nuevamente supera el valor de 1.0, y la exactitud no supera el 70 %. Sin embargo, la matriz de confusión muestra un comportamiento más claro que el octavo modelo en cuanto a la diagonal de aciertos. Esto último tiene que ver con el incremento de imágenes obtenidas de los 10 sujetos de prueba.

# CAPÍTULO 10

# Simulaciones

## 10.1. Interfaz

### 10.1.1. Prueba en tiempo real

Para las primeras pruebas se utilizaron los 4 modelos exportados, la cámara integrada y un bucle infinito. En este caso es utilizaron las siguientes librerías *CV2*, *TensorFlow* y *Numpy*. El primer paso de la configuración fue cargar el modelo previamente exportado. Luego, se realiza la inicialización de la cámara integrada por medio de *OpenCV*, y se ajusta el tamaño de la imagen a obtener por medio de comandos de la librería *CV2*. Ahora bien, en el bucle infinito se realiza una captura de la imagen obtenida por la cámara, la cual se guarda como una matrices de píxeles. Luego, se cambia el canal de BGR a RGB, se normaliza la matriz al dividirla entre 255, y se cambia la matriz de tipo flotante a entero. Después, por medio de la librería *TensorFlow* se utiliza la función Predecir, la cual devuelve una vector con el porcentaje de similitud predecido para cada clase. Finalmente, se aplica la función de *Numpy* para obtener el valor más máximo dentro de dicho vector de predicciones y dicho valor se imprime en la consola inicialmente.

### 10.1.2. Bloques de funciones

La primera versión de la interfaz creada, utiliza 3 grandes grupos de códigos. El primer grupo se encarga de capturar fotos y clasificarlas según la etiqueta que se necesite utilizar, en este caso se trabaja con 4. En esta ventada se presentan instrucciones en forma de lista para que el usuario pueda identificar y capturar las fotografías, por medio de una cuadrícula de selección. Luego, el segundo bloque cuenta con las opciones necesarias para preparar las imágenes de tal manera que puedan ser interpretadas por los modelos de programación. Por último, está el bloque de prueba en vivo, esta utiliza el código descrito en la sección "Prueba en tiempo real". Además, es importante resaltar que todos estos bloques funcionan con códigos los códigos utilizando en el capítulo "Algoritmos" dado que estos fueron construidos con

programación orientada a objetos. Por último, para la creación de esta interfaz se utilizaron las librerías *Tkinter* y *CustomTkinter* respectivamente.

## 10.2. Consola (Primer grupo de modelos)

En el caso de los primeros modelos realizados, la experiencia fue pésima, dado que el modelo predecía en su mayoría las posiciones inferiores, tanto para los modelos con 9 y 6 clases. Se intentó realizar dicho proceso de movimiento a diferentes distancias de la cámara pero el resultado fue el mismo. Para visualizar los resultados de estos modelos, se utilizó el código de prueba en tiempo real, imprimiendo valor de la predicción del modelo en la consola, como estaba previsto inicialmente

## 10.3. Turtle (Segundo grupo de modelos)

En cuanto a los modelos de 4 y 3 clases, el resultado depende de la distancia de separación del rostro del usuario y de la cámara. A más de 30 cm de distancia, las predicciones son bastante malas, dado que muchas veces se la predicción es errónea en cuanto al lado de la orientación de la cabeza. Sin embargo, al realizar pruebas justo a 30 cm de la cámara, tanto para el modelo de 3 y 4 clases, el resultado fue satisfactorio. Esto puede indicar que falta entrenar al modelo con diferentes tipos de fondos, dado que cuando la cámara capta en su mayoría el rostro de la persona, las predicciones son buenas. Además, se realizó la prueba a más de 30 cm en diferentes espacios de trabajo en la casa de Gerardo Fuentes, y los resultados no fueron buenos, hasta que se realizó el acercamiento nuevamente. En este caso, ya se contaba con la primera versión de la interfaz que es capaz de capturar y manipular imágenes, así como graficar por medio de la librería *Turtle* de *Python* el movimiento de un cursor, así como su trayectoria.

## 10.4. Webots (Tercer grupo de modelos)

En esta simulación fue necesario utilizar el software *Webots*, crear un mundo simple, que contuviera un piso rectangular y el robot a utilizar. En este caso se optó por el modelo del robot *E-puck*, dado que es un agente robótico móvil de dos ruedas, muy similar a una silla de ruedas. Además, se configuró el robot para utilizar un controlador externo, en vez de la opción por defecto, que permite utilizar un archivo incluido en la interfaz del software. La selección de un controlador externo fue realizada, dado que se requiere utilizar tanto las librerías del robot, como las que se han implementado para el manejo de datos y la predicción de clases en tiempo real. También es importante resaltar, que el código utilizado para manejar la simulación en *Webots*, aún no se ha incorporado a la interfaz.

En estas simulaciones, el séptimo y octavo modelos funcionaron de manera similar. En esta ocasión, la predicción de las posiciones funcionaba en un rango mayor de distancias, sin embargo existía una tendencia a predecir posiciones centrales de la cabeza como rotaciones a la derecha. Sin embargo, este último problema parecía no aparecer cuando el usuario se

posicionaba con un desfase respecto de la cámara hacia la izquierda. Por último, el noveno modelo funciona en un mayor rango de posiciones, distancias y fondos, aunque requiere utilizar los marcadores faciales, incluso así, llegando a presentar los problemas mencionados con el séptimo y octavo modelos, pero con menor error. Además, estos resultados fueron obtenidos al ser Gerardo Fuentes (autor) el usuario, dado que con usuarios nuevos, el resultado aún no es óptimo. Todo esto se consiguió, con una exactitud menor al 70 % que se observó en las gráficas de dicho modelo en la sección de " Tercer grupo de modelos de aprendizaje" en el capítulo "Algoritmos".

# CAPÍTULO 11

## Pruebas físicas

# CAPÍTULO 12

# Conclusiones

# CAPÍTULO 13

## Recomendaciones

# CAPÍTULO 14

# Bibliografía


[1] R. Mehta y V. Jain, "Computer Vision Based Prototype Model of Face Gesture Controlled Vehicle," jul. de 2021, págs. 313-317. DOI: 10.1109/CCICT53244.2021.00065.

[2] S.-W. Lee, *Automatic gesture recognition for intelligent human-robot interaction ...* Mayo de 2006. dirección: https://ieeexplore.ieee.org/document/1613091.

[3] S. Sharma, S. Jain y Khushboo, "A Static Hand Gesture and Face Recognition System for Blind People," en *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2019, págs. 534-539. DOI: 10.1109/SPIN.2019.8711706.

[4] R. T. Bankar y S. S. Salankar, "Head Gesture Recognition System Using Gesture Cam," en *2015 Fifth International Conference on Communication Systems and Network Technologies*, 2015, págs. 535-538. DOI: 10.1109/CSNT.2015.81.

[5] L. A. Rivera, G. N. DeSouza y L. D. Franklin, "Control of a wheelchair using an adaptive K-Means clustering of head poses," en *2013 IEEE Symposium on Computational Intelligence in Rehabilitation and Assistive Technologies (CIRAT)*, 2013, págs. 24-31. DOI: 10.1109/CIRAT.2013.6613819.

[6] X. Wang, J. Jang, Y. Wei, L. Kang e Y. Gao, *Research on Gesture Recognition Method Based on Computer Vision Technology*, nov. de 2018. dirección: https://www.matec-conferences.org/articles/matecconf/abs/2018/91/matecconf_eitce2018_03042/matecconf_eitce2018_03042.html.

[7] R. Szeliski, *Computer vision: Algorithms and applications*. Springer, 2023.

[8] IBM, *What is Computer Vision?* Dirección: https://www.ibm.com/topics/computer-vision.

[9] A. Krizhevsky, I. Sutskever y G. E. Hinton, *ImageNet classification with deep convolutional Neural Networks*. dirección: https://www.researchgate.net/publication/267960550_ImageNet_Classification_with_Deep_Convolutional_Neural_Networks.

[10] Dirección: https://www.mathworks.com/discovery/object-detection.html.

[11] V. Lepetit, F. Moreno-Noguer y P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, feb. de 2009. DOI: 10.1007/s11263-008-0152-6.

[12] M. Walia, *What is object tracking in computer vision?* Dic. de 2022. dirección: https://blog.roboflow.com/what-is-object-tracking-computer-vision/.

[13] A. Acharya, *The Complete Guide to Object Tracking [tutorial]*, oct. de 2023. dirección: https://encord.com/blog/object-tracking-guide/.

[14] N. Barla, *The Complete Guide to Object Tracking [+V7 tutorial]*, nov. de 2021. dirección: https://www.v7labs.com/blog/object-tracking-guide.

[15] N. Klingler, *Object tracking in computer vision (2024 guide)*, nov. de 2023. dirección: https://viso.ai/deep-learning/object-tracking/.

[16] M. RadhaKrishna et al, *A review on Image Processing Sensor*, 2021. dirección: https://iopscience.iop.org/article/10.1088/1742-6596/1714/1/012055.

[17] X. W. et al, *The evolution of LIDAR and its application in high precision ...* dirección: https://iopscience.iop.org/article/10.1088/1755-1315/502/1/012008.

[18] C. Atwell, *What is a radar sensor?* Mar. de 2021. dirección: https://www.fierceelectronics.com/sensors/what-a-radar-sensor.

[19] R. Ingle, *How do radar sensors work?* Ago. de 2023. dirección: https://www.azosensors.com/article.aspx?ArticleID=2854.

[20] L. Li, *Time-of-flight camera – an introduction - texas instruments india*, 2014. dirección: https://www.ti.com/lit/wp/sloa190b/sloa190b.pdf.

[21] M. Andersen, T. Jensen, P. Lisouski et al., "Kinect Depth Sensor Evaluation for Computer Vision Applications," *Technical Report Electronics and Computer Engineering*, vol. 1, n.º 6, sep. de 2012. dirección: https://tidsskrift.dk/ece/article/view/21221.

[22] N. Mahamkali y V. Ayyasamy, "OpenCV for Computer Vision Applications," mar. de 2015.

[23] M. Abadi, P. Barham, J. Chen et al., "TensorFlow: A system for large-scale machine learning," mayo de 2016.

[24] D. Kirk, "NVIDIA CUDA software and GPU parallel computing architecture," vol. 7, oct. de 2007, págs. 103-104. DOI: 10.1145/1296907.1296909.

[25] Dirección: https://www.mathworks.com/discovery/computer-vision.html.

[26] Dirección: https://caffe.berkeleyvision.org/.

[27] N. Gourier, D. Hall y J. L. Crowley, *Estimating Face orientation from Robust Detection of Salient Facial Structures*. dirección: http://crowley-coutaz.fr/jlc/papers/Pointing04-Gourier.pdf.

# CAPÍTULO 15

# Anexos