DESARROLLO DE DISPENSADOR DE ALIMENTO PARA MASCOTAS CON DETECCIÓN FACIAL

MIGUEL ÁNGEL CARMONA TORRES

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO
FACULTAD DE INGENIERÍA
TECNOLOGÍA EN ELECTRÓNICA
MEDELLÍN
2023

DESARROLLO DE DISPENSADOR DE ALIMENTO PARA MASCOTAS CON DETECCIÓN FACIAL

MIGUEL ÁNGEL CARMONA TORRES

Trabajo de grado para optar al título de Tecnólogo en Electrónica

Asesor Técnico
Sergio Hernando Ruiz Obando
Magíster en Tecnologías Digitales Aplicadas a la Educación

Asesor Metodológico
Diego Hernando Orozco Gómez
Magíster en Ingeniería – Automatización Industrial

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO
FACULTAD DE INGENIERÍA
TECNOLOGÍA EN ELECTRÓNICA
MEDELLÍN
2023

Agradecimientos

En este significativo momento de culminación académica, deseo expresar mi más profundo agradecimiento a Dios, fuente inagotable de fortaleza y guía, por iluminar mi camino durante este arduo proceso. A mis padres, cuyo amor, apoyo incondicional y sacrificios han sido la base de mi trayectoria académica, les estoy eternamente agradecido. Agradezco sinceramente a mis respetados profesores, cuyas exigencias y orientaciones no solo desafiaron mis límites, sino que también fueron esenciales para alcanzar los estándares académicos establecidos. Cada lección, evaluación y corrección contribuyeron significativamente a mi desarrollo y éxito en este proyecto. Gracias a todos quienes, de diversas maneras, fueron parte fundamental de este viaje académico.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de anexos

Pág.

Resumen

DESARROLLO DE DISPENSADOR DE ALIMENTO PARA MASCOTAS CON DETECCIÓN FACIAL

MIGUEL ÁNGEL CARMONA TORRES

La concepción y desarrollo de un dispensador automatizado para mascotas con reconocimiento facial surge de la necesidad de abordar la carga financiera asociada con la tenencia de animales y promover prácticas responsables de cuidado animal. Integrando tecnologías avanzadas, el proceso de creación incluyó la implementación del reconocimiento facial para garantizar la seguridad alimentaria y monitorear hábitos, brindando una herramienta valiosa para el cuidado de la salud animal. Este dispositivo no solo ha demostrado eficiencia en la entrega de comidas, sino que también ha impactado positivamente la comunidad al aumentar la adopción de mascotas, empoderando a más individuos para asumir la responsabilidad de cuidar seres vivos. En última instancia, la creación de este dispensador automatizado no solo resuelve problemas prácticos, sino que también contribuye a formar comunidades más compasivas y comprometidas con el bienestar de los animales, construyendo un futuro donde la tenencia responsable de mascotas sea más accesible y se fortalezca el vínculo entre las personas y sus mascotas.

Palabras claves: dispensador, mascota, reconocimiento, red neuronal

Abstract

DEVELOPMENT OF AUTOMATED PET FOOD DISPENSER WITH FACIAL RECOGNITION

MIGUEL ÁNGEL CARMONA TORRES

The conception and development of an automated pet food dispenser with facial recognition arise from the need to address the financial burden associated with pet ownership and promote responsible animal care practices. By integrating advanced technologies, the creation process involved the implementation of facial recognition to ensure food safety and monitor habits, providing a valuable tool for animal health care. This device has not only demonstrated efficiency in meal delivery but has also positively impacted the community by increasing pet adoption, empowering more individuals to take responsibility for caring for living beings. Ultimately, the creation of this automated dispenser not only solves practical problems but also contributes to the formation of more compassionate and committed communities to animal welfare, building a future where responsible pet ownership is more accessible, and the bond between people and their pets is strengthened.

Keywords: dispenser, pet, recognition, neural network

Glosario

Ácido fólico: es una vitamina B, el cuerpo lo usa para producir células nuevas. Este ácido es la forma sintética del folato que se usa en los suplementos y en los alimentos fortificados, como arroz, pastas, pan y algunos cereales en el desayuno.

Acido pantoténico: es una vitamina hidrosoluble que a su vez es un precursor en la síntesis de la coenzima A. La coenzima A es esencial para muchas reacciones bioquímicas que sostienen la vida.

Aprendizaje de máquina: rama de la Inteligencia Artificial que se encarga del descubrimiento de patrones y relaciones en los datos para hacer predicciones o tomar decisiones.

Aprendizaje profundo: técnica del aprendizaje de máquina que hace uso de redes neuronales profundas para resolver problemas de clasificación.

Clasificación: técnica del aprendizaje de máquina en la que, a partir de miles y miles de ejemplos con atributos y la categoría en la que está clasificado cada ejemplo, encuentra patrones y relaciones para clasificar nuevos ejemplos.

Código G: del inglés G-Code, es un lenguaje de programación en formato texto. Se ha popularizado mucho gracias a la impresión 3D, debido a que el G-Code especifica dónde posicionar el extrusor, en horizontal y vertical, cuánto calentarlo y cómo disponer el filamento entre otras cosas.

e Passport: pasaporte electrónico o pasaporte biométrico. La característica principal de éste es la incorporación (siguiendo la norma ICAO, International Civil Aviation Organization) de un chip sin contacto, protegido por una lámina de seguridad de policarbonato. El chip sin contacto puede estar embebido en una de sus cubiertas, en una página de datos, en el centro de la libreta o bien en una página cosida por separado, dependiendo del país.

Fresadora CNC: es una máquina que permite automatizar la producción de piezas, de una forma eficaz, segura y económica. En un periodo de tiempo menor se producen un mayor volumen de piezas con una garantía total de perfección. Estas máquinas permiten realizar el mismo proceso un innumerable número de veces sin perder precisión.

Home assistant: o asistente de hogar en español, es un software de automatización de hogar, gratuito, de código abierto y local.

IoT: agrupación e interconexión de dispositivos y objetos a través de una red (bien sea privada o Internet, la red de redes), dónde todos ellos podrían ser visibles e interaccionar.

PEEK: es un polímero termoplástico orgánico perteneciente a la familia de las polietercetonas, también llamadas rey de los polímeros, con una combinación de propiedades mecánicas, térmicas y químicas muy elevadas.

PEI: es un termoplástico amorfo con gran resistencia mecánica y rigidez.

PLA: ácido poli láctico (PLA). Se produce a partir de materias primas de origen natural y renovables como lo puede ser el maíz. Forma parte del grupo de los poliésteres como un polímero sintético.

Retro-propagación: algoritmo iterativo que, luego de cada repetición, recorre hacia atrás una red neuronal profunda ajustando los pesos de las conexiones en función de la contribución al error.

Sistema domótico: es un sistema que utiliza distintos sensores para recoger información, procesarla y emitir órdenes a actuadores específicos aplicados al hogar.

Introducción

Este proyecto se centra en el desarrollo de un dispensador de alimentos para gatos con tecnología de reconocimiento facial, diseñado con el propósito específico de prevenir la sobrealimentación. A través de un análisis meticuloso, se ha priorizado la selección de componentes de bajo costo, garantizando eficiencia y accesibilidad en el diseño del dispositivo. La inclusión del reconocimiento facial permite una identificación precisa de cada gato, permitiendo un control personalizado en el suministro de alimentos. Este dispensador automatizado se proyecta como una innovación destinada a mejorar la calidad de vida de las mascotas, abordando de manera integral el desafío de la sobrealimentación y subrayando la relevancia de la tecnología en la promoción del bienestar animal.

El método empleado en este trabajo es de naturaleza inductiva, siguiendo un enfoque metodológico que abarca varias etapas. Inicialmente, se lleva a cabo la investigación y selección de componentes de hardware, priorizando la eficiencia y accesibilidad económica. Posteriormente, se programa un microcontrolador para implementar funciones de dispensación personalizadas. La integración de la tecnología de reconocimiento facial y una Red Neuronal es esencial en la identificación de las mascotas y la determinación de sus necesidades alimenticias. El proceso se complementa con pruebas exhaustivas y ajustes para garantizar el correcto funcionamiento y la seguridad del sistema. Se incorpora un análisis económico para evaluar la viabilidad del producto y se adopta un enfoque ético que considera el bienestar de las mascotas, asegurando así el cumplimiento de los objetivos y el desarrollo de un dispensador automatizado eficaz y accesible.

A pesar de los avances significativos, este trabajo presenta ciertas limitaciones. Se identifican desafíos en la adaptabilidad del reconocimiento facial a diversas condiciones, como variaciones en la iluminación y ángulos de captura. Además, se reconoce la necesidad de explorar mejoras potenciales en la identificación contextual del cuerpo del gato. Estas limitaciones, aunque reconocidas, sirven como motivación para futuras investigaciones y refinamientos en la tecnología del dispensador, buscando continuamente la optimización de su desempeño y la maximización de su impacto en el bienestar animal.

1. Planteamiento del problema

1.1 Descripción

Cuando las personas optan por adquirir una mascota, a menudo pasan por alto la importancia de proporcionar la atención adecuada en cuanto a su alimentación. Suministrar alimentos de calidad en horarios regulares se convierte en un aspecto fundamental para mantener una nutrición equilibrada y garantizar comidas equitativas. Al igual que los humanos, las mascotas requieren una dieta adecuada para prevenir el deterioro de su calidad de vida y evitar enfermedades como la toxicidad renal, enfermedades cardiovasculares, diarrea, vómito y gastritis, entre otras afecciones que pueden derivarse de una mala digestión. Estas enfermedades, en última instancia, pueden poner en riesgo la vida de los fieles compañeros, lo que lamentablemente contribuye al aumento del índice de abandono, exacerbando el problema de la sobrepoblación de animales en las calles y generando una preocupante problemática de salud pública.

A pesar de que existen dispositivos automatizados en el mercado diseñados para la administración de alimentos a las mascotas, no todas las personas pueden acceder a ellos debido a sus elevados costos. Esto hace que la inversión en tales dispositivos no siempre justifique sus beneficios, ya que los grandes fabricantes a menudo ofrecen un único modelo que puede resultar limitante en términos de adaptación a diferentes entornos o necesidades específicas de control y automatización en la alimentación de las mascotas.

La falta de tiempo y disciplina en el estilo de vida de las personas que tienen mascotas a menudo impide que se le dé la debida importancia a la alimentación de sus animales de compañía. Por este motivo, el presente proyecto se plantea principalmente con el objetivo de mejorar tanto la calidad de vida de las mascotas como la de sus dueños. Con demasiada frecuencia, abandonar a un animal en la calle se convierte en la única opción viable cuando ha desarrollado una enfermedad irreversible, principalmente debido a los costos prohibitivos que pueden conllevar los tratamientos veterinarios en la actualidad. Motivados por esta problemática, la iniciativa busca identificar soluciones electrónicas posibles que permitan asistir a los animales

abandonados a través de la distribución de alimentos, la medición precisa de las cantidades suministradas, además de otras que se pueden derivar a partir de esta como la promoción de campañas de vacunación y esterilización, y el fomento de la adopción de mascotas.

1.2 Formulación

¿Cómo la incorporación de un sistema de detección facial en la alimentación de mascotas podría mejorar la experiencia de cuidado de las mascotas mediante una gestión personalizada y eficiente de su alimentación?

2. Justificación

El objetivo primordial de este proyecto es el desarrollo de un dispensador automatizado diseñado para ofrecer una solución a la alimentación de las mascotas, especialmente en situaciones donde sus dueños enfrentan limitaciones de tiempo para alimentarlas de manera regular. La atención de una alimentación adecuada puede tener un impacto negativo en la salud de los animales, lo que puede dar lugar a la desnutrición y el surgimiento de enfermedades. Este proyecto busca no solo garantizar la alimentación adecuada de las mascotas, sino también controlar la proliferación de casos de abandono de animales, un problema agravado por las consecuencias de la pandemia de COVID-19 que aún afecta a la sociedad. Un dispensador automatizado se presenta como una solución práctica no solo para los dueños de mascotas con horarios ocupados o impredecibles, sino también como una herramienta para mejorar la salud. y el bienestar de las mascotas, asegurando su alimentación incluso en momentos en que sus dueños no pueden estar presentes.

En Colombia, la Asociación Colombiana Protectora de Animales estima que alrededor de 300.000 perros y gatos son abandonados cada año, lo que contribuye significativamente a la problemática de la sobrepoblación de animales en situación de calle en muchas ciudades del país. A nivel mundial, la Organización Mundial de la Salud (OMS) estima que existen más de 200 millones de perros y 600 millones de gatos en situación de calle. Estos animales a menudo enfrentan condiciones de hambre, enfermedades, abuso y diversos peligros, lo que deteriora su salud y bienestar. Es esencial recordar que el abandono de animales representa una práctica inhumana y cruel que implica sufrimiento innecesario y dolor. Toda la humanidad comparte la responsabilidad de cuidar y proteger a los animales, promoviendo como la adopción y la esterilización para prevenir la sobrepoblación y el abandono de mascotas (Noticias HSB, 2022).

En la actualidad, diversos factores impactan en la salud de las mascotas, factores que a menudo pasan desapercibidos. Por ejemplo, la altura del plato de comida puede generar tensión en el cuello y los hombros del animal en caso de no ser adecuada, lo que aumenta el riesgo de problemas de salud. Asimismo, la cantidad de alimento proporcionado debe ser considerada cuidadosamente en función de la edad y el peso del animal, ya que una dieta equilibrada y

apropiada en términos de cantidad es fundamental para su salud y bienestar a largo plazo. La cantidad de alimento requerida varía según la especie, raza, edad, nivel de actividad, estado de salud y otros factores individuales de cada mascota. Un exceso de alimentación puede llevar a problemas de salud como la obesidad, enfermedades cardíacas, diabetes y trastornos articulares, mientras que una dieta deficiente en nutrientes puede tener un impacto negativo en el desarrollo y la salud a largo plazo de la mascota, generando una serie de problemas.

3. Objetivos

3.1 Objetivo general

Desarrollar un dispensador automatizado de bajo costo para mascotas con el propósito del mejoramiento en la calidad de vida de los animales domésticos y la simplificación la rutina de los dueños que enfrentan limitaciones de tiempo, al mismo tiempo que se promueve el bienestar y salud de las mascotas.

3.2 Objetivos específicos

Diseñar un sistema de hardware eficiente y de bajo costo que cumpla con los requisitos técnicos necesarios para la operación del dispensador automatizado.

Realizar el entrenamiento de una red neuronal que permita la identificación correcta entre dos gatos.

Verificar el desempeño del sistema de reconocimiento de tal forma que pueda ser incorporado en el dispensador.

4. Marco teórico

4.1 Nutrición correcta a los gatos

Los gatos necesitan los siguientes nutrientes para mantenerse sanos:

Proteína: la proteína es esencial para los gatos. La proteína ayuda a promover el crecimiento de los músculos magros.

Grasas: la grasa se encuentra a menudo en forma de ácidos grasos como el omega-3 y omega-6. De manera similar a los humanos, el exceso de grasa puede llevar al aumento de peso, pero la cantidad correcta de grasa puede ayudar a mantener sanos su piel y pelaje.

Fibra: los gatos necesitan la cantidad correcta de fibra encontrada en fuentes como el arroz integral para favorecer una salud digestiva adecuada y prevenir el exceso de gases y otros problemas gastrointestinales.

Agua: los gatos necesitan bastante agua fresca y limpia diariamente para mantenerse sanos día a día.

Los gatos también necesitan un buen número de vitaminas y minerales en su alimento. En la vida silvestre, estas vitaminas y minerales están disponibles en sus presas. Sin embargo, un gato doméstico debe recibir las vitaminas y minerales adecuados por medio de su alimento. Como las personas, los gatos necesitan vitaminas A, E, K y B. También se necesitan ácidos fólico y pantoténico para un sano crecimiento y mantenimiento. De acuerdo con PetMD, los gatos producen bastante vitamina C por sí mismos, a diferencia de sus dueños, y no necesitan complementos de vitamina C para mantenerse sanos. Adicionalmente, los gatos necesitan otros minerales como el calcio, fósforo y yodo. La taurina es un aminoácido esencial necesario para los gatos en su nutrición diaria. Estas vitaminas, minerales y nutrientes deben darse al gato en el alimento que se le sirva, no es necesario el uso de complementos. Los gatos pueden vivir con un

plan de alimentación a base de alimento seco solamente, recibiendo una fuente completa de nutrición balanceada (Published, 2016).

Figura 1. Dispensador convencional para gatos
Fuente: extraído de https://www.hillspet.com.mx/cat-care/nutrition-feeding/choosing-healthy-cat-food

Los gatos pequeños tienen estómagos de menor tamaño, así que se debe administrar tres comidas pequeñas hasta los 6 meses de edad en vez de darle acceso libre al alimento las 24 horas, porque puede conducir a malos hábitos de alimentación en la edad adulta. Las pequeñas bocas y dientes de los gatos son otra razón importante para no darle un alimento para adultos todavía; cada uno de esos bocados grandes puede representar peligro de asfixia (Published, 2016).

4.2 Reconocimiento facial

El reconocimiento facial es una manera de identificar o confirmar la identidad de una persona mediante su rostro. Los sistemas de reconocimiento facial se pueden utilizar para identificar a las personas en fotos, videos o en tiempo real. El reconocimiento facial es una categoría de seguridad biométrica. Otras formas de software biométrico incluyen el reconocimiento de voz, el reconocimiento de huellas digitales y el reconocimiento de retina o iris. La tecnología se utiliza principalmente para la protección y las fuerzas de seguridad, aunque hay un creciente interés en otras áreas de uso.

Muchas personas están familiarizadas con la tecnología de reconocimiento facial a través de FaceID que se usa para desbloquear iPhone (sin embargo, este es solo uno de los usos de esta tecnología). Por lo general, el reconocimiento facial no depende de una base de datos masiva de fotos para determinar la identidad de una persona; simplemente identifica y reconoce a un individuo como el único propietario del dispositivo, a la vez que limita el acceso a otros.

Además de desbloquear teléfonos, el reconocimiento facial funciona comparando los rostros de las personas que pasan frente a cámaras especiales con las imágenes de personas en una lista de control. Las listas de control pueden contener fotografías de cualquier persona, incluidas aquellas de las que no se sospecha ningún acto ilícito, y las imágenes pueden provenir de cualquier lugar, incluso de cuentas de redes sociales. Los sistemas de tecnología facial pueden variar, pero en general tienden a funcionar de la siguiente manera:

Paso 1. Reconocimiento facial. La cámara detecta y ubica la imagen de un rostro, ya sea de forma independiente o como parte de una muchedumbre. La imagen puede mostrar a la persona de frente o de perfil.

Paso2. Análisis facial. A continuación, se captura y analiza una imagen del rostro. La mayor parte de la tecnología de reconocimiento facial depende de imágenes 2D en lugar de 3D, ya que se puede comparar de manera más fácil una imagen 2D con las fotos públicas o las de una base de datos. El software lee la geometría del rostro. Los factores clave incluyen la distancia entre los ojos, la profundidad de las cuencas de los ojos, la distancia desde la frente hasta el mentón, la forma de los pómulos y el contorno de los labios, las orejas y el mentón. El objetivo es identificar los puntos de referencia faciales que son clave para distinguir un rostro.

Paso 3. Conversión de la imagen a datos. El proceso de captura de rostro transforma la información analógica (un rostro) en un conjunto de información digital (datos) basado en los rasgos faciales de la persona. Básicamente, el análisis del rostro se convierte en una fórmula matemática. El código numérico se denomina huella facial. De la misma manera en que las huellas dactilares son únicas, cada persona tiene su propia huella facial.

Paso 4. Búsqueda de una coincidencia. La huella facial se compara con una base de datos de otros rostros conocidos. Por ejemplo, el FBI tiene acceso hasta 650 millones de fotos, de diversas bases de datos estatales. En Facebook, cualquier foto etiquetada con el nombre de una persona se convierte en parte de la base de datos de Facebook, que también puede usarse para el reconocimiento facial. Si la huella facial coincide con una imagen en una base de datos de reconocimiento facial, entonces se realizará una determinación.

De todas las mediciones biométricas, el reconocimiento facial se considera el más natural. Esto sigue una lógica intuitiva, ya que normalmente el reconocimiento de una persona se realiza mirando caras, en lugar de las huellas digitales y los iris. Se estima que, periódicamente, más de la mitad de la población mundial se ve afectada por la tecnología de reconocimiento facial.

La tecnología de reconocimiento se utiliza para una variedad de propósitos. Entre estas, se incluyen las siguientes:

Desbloqueo de teléfonos. Varios teléfonos, incluidos los iPhone más recientes, usan el reconocimiento facial para desbloquear el dispositivo. La tecnología ofrece una potente manera de proteger los datos personales y garantiza que los datos confidenciales permanezcan inaccesibles si roban el teléfono. Apple dice que la posibilidad de que una cara aleatoria desbloquee el teléfono es de aproximadamente una en 1millón.

Fuerzas de seguridad. Las fuerzas de seguridad utilizan el reconocimiento facial de forma cotidiana. El uso de esta tecnología está aumentando entre las agencias de las fuerzas de seguridad en Estados Unidos, y lo mismo ocurre en otros países. La policía recopila las fotos de los arrestados y las compara con las bases de datos locales, estatales y federales de reconocimiento facial. Una vez que se toma la foto de un arrestado, esta se agrega a las bases de datos para que se escanee cada vez que la policía realice otra búsqueda criminal. Además, el reconocimiento facial móvil permite a los agentes de policía utilizar teléfonos inteligentes, tabletas u otros dispositivos portátiles para tomar una foto de un conductor o peatón en terreno y compararla de inmediato con una o más bases de datos de reconocimiento facial con el fin de intentar una identificación.

Control de aeropuertos y fronteras. El reconocimiento facial se ha convertido en una escena familiar en muchos aeropuertos de todo el mundo. Cada vez más viajeros tienen pasaportes biométricos, lo que les permite saltarse las largas filas y, en su lugar, desplazarse a través de un control automatizado de ePassport para llegar más rápido a la puerta de embarque. Además de reducir los tiempos de espera, el reconocimiento facial permite a los aeropuertos mejorar la seguridad. El Departamento de Seguridad Nacional de los Estados Unidos predice que, para el año 2023, el reconocimiento facial se utilizará en un 97 % de los viajeros. Además de los aeropuertos y cruces fronterizos, la tecnología se utiliza para mejorar la seguridad en eventos a gran escala, como los Juegos Olímpicos (Kaspersky, 2023).

4.2.1 Reconocimiento facial en mascotas. El uso del reconocimiento facial en animales ha demostrado tener mayor eficiencia que en los humanos. En la mayoría de los casos, se utiliza para mejorar sus vidas con el propósito de preservar algunas especies y también por razones comerciales. La aplicación GoGo Chicken, utiliza el reconocimiento facial para distinguir a los pollos, con el fin de que los consumidores puedan comprobar fácilmente el lugar de nacimiento, la alimentación y la información sanitaria del animal recién comprado, sin tener que recurrir a etiquetas o chips (Barbieri, 2018).

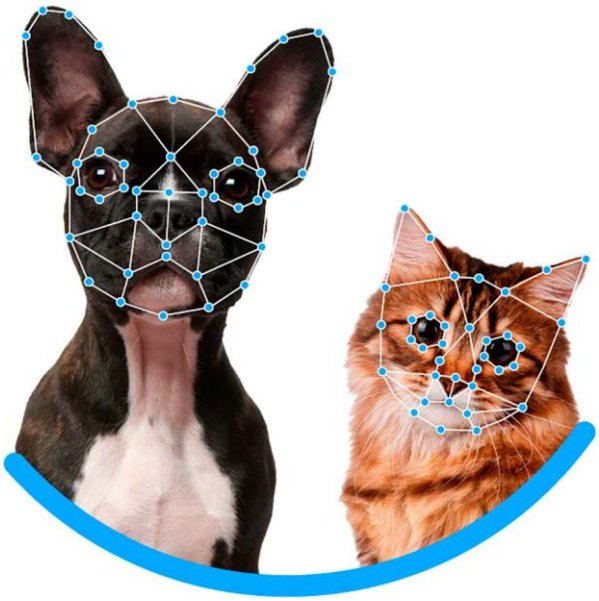

Figura 2. Reconocimiento facial de mascotas
Fuente: extraído de https://www.doogweb.es/2019/11/22/mascofinder-reconocimiento-facial-perros-perdidos/

4.3 Redes neuronales

Las redes neuronales son el avance más reciente del aprendizaje de máquina, la rama de la Inteligencia Artificial que se encarga del descubrimiento de patrones y relaciones entre los datos para hacer predicciones o tomar decisiones. Desde el inicio de su auge en 2012, las redes neuronales están detrás de sistemas avanzados de reconocimiento facial, la creación de textos por computadora, la conducción de los vehículos autónomos. Parte del gran éxito de las redes neuronales se debe a que pueden utilizarse en actividades que parecieran ser muy diferentes, pero que en realidad corresponden al mismo tipo de problema, llamado clasificación en el aprendizaje de máquina.

Hoy en día sistemas de aprendizaje profundo alcanzan resultados que superan los que un ser humano puede lograr en cada vez más actividades, además de incentivar la exploración permanente de aplicaciones en nuevos campos que están transformando industrias y economías. Esto gracias al uso de múltiples unidades de procesamiento gráfico, originalmente diseñadas para juegos, aunque resultaron muy efectivas para hacer estos cálculos, modelos de redes en las que neuronas se conectan con otras de la misma capa o de capas anteriores, así como millones de ejemplos para el aprendizaje (Inteligencia Futura, 2020).

Figura 3. Etapas del reconocimiento facial de mascotas
Fuente: extraído de
http://sedici.unlp.edu.ar/bitstream/handle/10915/118496/Documento_completo.pdf?sequence=1&isAllowed=y

4.4 ESP32-CAM

La ESP32-CAM es un dispositivo que puede llamarse un todo en uno, incluye pines GPIO, conectividad Wifi y Bluetooth. Lleva integrado una pequeña cámara de video y una conexión

para una tarjeta MicroSD donde se puede almacenar fotos o videos. Su bajo precio permite que sea un dispositivo muy utilizado en IoT, además, conectarlo a Home Assistant y que forme parte de tu sistema domótico, es muy fácil. Todo esto con un peso de 20 gramos y unas pequeñas dimensiones. Las aplicaciones típicas son: toma de fotos, streaming de video, reconocimiento facial y detector de movimiento (Pascual, 2022).

Figura 4. Características de la ESP32-CAM
Fuente: extraído de https://programarfacil.com/esp32/esp32-cam/

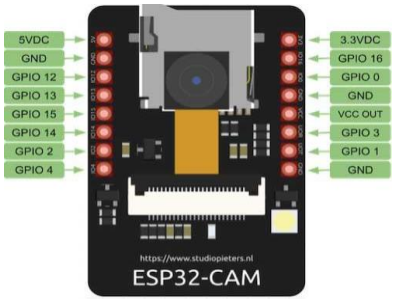

Figura 5. Esquema de la ESP32-CAM
Fuente: extraído de https://programarfacil.com/esp32/esp32-cam/

4.5 Impresión 3D

La impresión 3D es el proceso de creación de objetos mediante el depósito de capas de material unas sobre otras. La impresión 3D se denomina fabricación aditiva (AM) en lugar de los métodos sustractivos tradicionales, como el fresado CNC, cuando se utiliza para la producción industrial. Esta tecnología existe desde hace unas cuatro décadas, inventada a principios de los años 80. Aunque la impresión 3D empezó siendo una técnica lenta y costosa, los amplios avances tecnológicos han hecho que las tecnologías AM actuales sean más asequibles y rápidas que nunca.

Un modelo digital en 3D se corta en cientos de capas finas mediante un software específico para exportarlo en formato de código G. Este formato de impresión 3D es un lenguaje que la impresora 3D lee para saber con precisión cuándo y dónde depositar el material. Cada capa corresponde a la forma 2D exacta de una sección o rebanada del objeto. Por ejemplo, si se imprimiera en 3D una pirámide, la primera capa (la inferior) sería un cuadrado plano, y la última capa (en la parte superior) sería un pequeño punto. Las capas se imprimen consecutivamente en 3D de una en una hasta obtener el objeto completamente impreso (Dassault Systèmes, 2023).

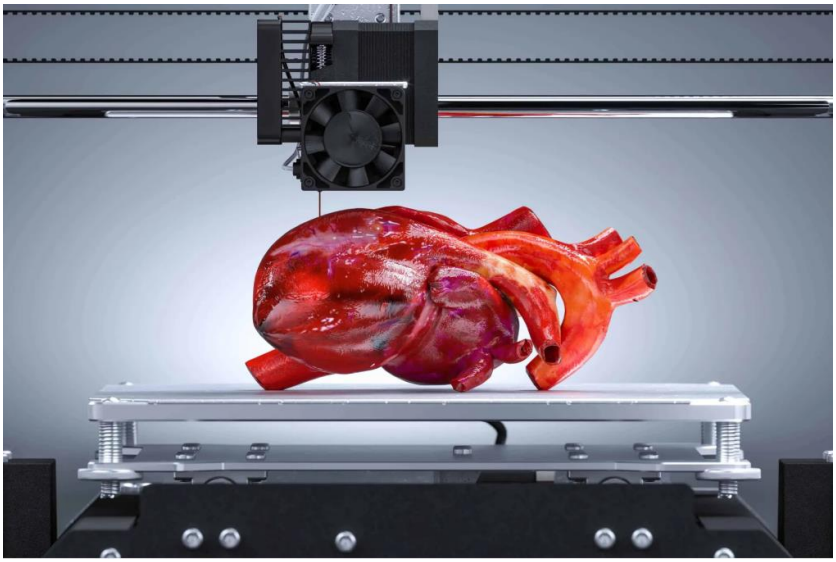

Figura 6. Impresión 3D
Fuente: extraído de https://www.3ds.com/es/make/guide/process/3d-printing

La impresión 3D ofrece un número considerable de ventajas, la más importante de las cuales es la capacidad de producir diseños muy complejos que serían imposibles de realizar de otro modo. Otra ventaja significativa de la impresión 3D es la velocidad. Aunque la impresión 3D de un objeto puede llevar horas o incluso días enteros, sigue siendo mucho más rápida que los métodos de producción habituales, como el moldeo por inyección.

La creación de prototipos que es uno de los usos más populares de la impresión 3D puede realizarse en la empresa con poco o ningún tiempo de espera, y las iteraciones del diseño pueden implementarse e imprimirse en el momento. Esta tecnología también ofrece muchas posibilidades de materiales de impresión en 3D.

Es posible imprimir en 3D con casi cualquier material. Los materiales de impresión 3D más comunes son los de base plástica, que van desde el PLA estándar hasta polímeros avanzados y muy resistentes como el PEEK o el PEI, y mucho más. Incluso es posible reforzar los termoplásticos con fibra de carbono o de vidrio. También están ganando popularidad algunos materiales de impresión 3D de nicho. Los científicos y biólogos están experimentando con la bioimpresión 3D, los chefs pueden probar la impresión 3D de alimentos y los contratistas están estudiando cada vez más la impresión 3D de hormigón (Dassault Systèmes, 2023).

5. Metodología

5.1 Tipo de proyecto

Este proyecto es de tipo aplicado, con un grado de profundidad exploratorio porque se desarrolla un producto o una tecnología utilizando componentes electrónicos de bajo costo que pueda contribuir a solucionar un problema específico como lo es la sobrealimentación en las mascotas.

5.2 Método

El método usado es el inductivo y se seguirá un enfoque metodológico que implica inicialmente la investigación y selección de componentes de hardware asequibles y eficientes, seguido de la programación de un microcontrolador para implementar las funciones de dispensación personalizadas. A continuación, se llevará a cabo la integración de tecnología de reconocimiento facial y Red Neuronal para identificar a las mascotas y determinar sus necesidades alimenticias. Posteriormente, se realizarán pruebas exhaustivas y ajustes en el sistema para garantizar su correcto funcionamiento y seguridad. Este proceso se complementará con un análisis económico para evaluar la viabilidad del producto y un enfoque ético que considere el bienestar de las mascotas, asegurando así la consecución de los objetivos y el desarrollo de un dispensador automatizado eficaz y accesible para mejorar la calidad de vida de las mascotas y sus dueños.

5.3 Instrumentos de recolección de información

5.3.1 Fuentes primarias. Sitios web de organizaciones de bienestar familiar, estudios de casos, informes y estadísticas gubernamentales.

5.3.2 Fuentes secundarias. Revistas especializadas, libros y artículos de investigaciones.

6. Resultados del proyecto

En este capítulo se detalla el desarrollo del proyecto enfocado en el desarrollo de un dispensador automatizado de alimentos para mascotas. Se presentan los detalles clave sobre el diseño de hardware eficiente y de bajo costo, la tecnología de reconocimiento facial y redes neuronales para garantizar la identificación segura de las mascotas, y las pruebas que se realizaron para corroborar el funcionamiento del modelo entrenado. Este capítulo brinda los aspectos técnicos y prácticos que respaldan la iniciativa para mejorar el bienestar de las mascotas.

6.1 Diseño del hardware

La implementación del dispensador parte de un diseño de hardware eficiente en términos de energía y costos, teniendo en cuenta los siguientes elementos:

- Diseños 3D del dispensador en formato STL
- 1 MG995 Servo
- 1 ESP32CAM
- 1 12V DC adaptador
- 1 DC Jack Conector de montaje en panel de conexión hembra
- 1 convertidor regulador de voltaje DC
- 2 rodamientos R3ZZ (12.7x4.98mm)
- M3 Tuercas de inserción por inyección
- M3 Barra roscada de rosca completa, sujetadores de acero inoxidable
- M3 Tuerca hexagonal de acero inoxidable
- Soldador
- Estaño
- Arduino

La compañía Electronoobs (Electronoobs, 2021) provee los diseños para la impresión 3D de las piezas que hacen parte del dispensador. Es posible diseñar las piezas para que el plato de

comida quede a una altura adecuada para la mascota. Al terminar las impresiones se deben quitar los soportes y lijar un poco donde sea necesario para el ensamble del dispensador.

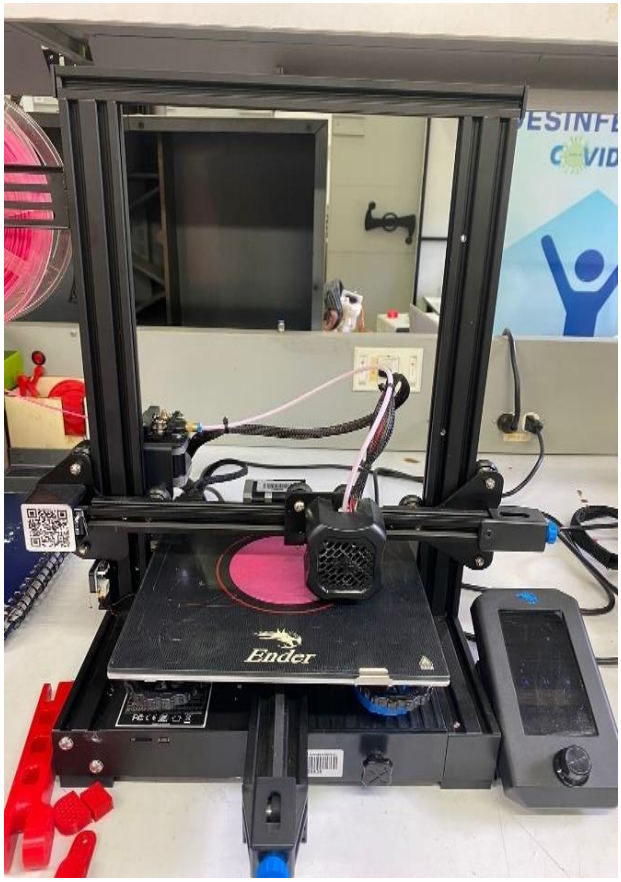

Figura 7. Proceso de impresión de las piezas requeridas del dispensador
Fuente: Laboratorio de Mecatrónica, Institución Universitaria Pascual Bravo

Todas las piezas fueron impresas en el laboratorio de Mecatrónica de la Institución Universitaria Pascual Bravo con material PLA. Se usaron dos perímetros, 20% de relleno, altura de capa de 0,25mm y boquilla de 0,4 mm. Esto se puede evidenciar en la Figura 8.

Debido a que se utilizó un servomotor MG995, el cual suele girar por defecto 180 grados y se desea que gire sin parar, se le hizo una modificación para que gire 360 grados. Si se abre cualquier servomotor se encontrará algún tipo de feedback que suele ser un potenciómetro. Para lograr que el motor del servo siempre este en posición media se requirió remplazar este potenciómetro con resistencias de valor fijo. Por tanto, se procedió a soldar dos resistencias de 4.7 KΩ en la PCB de control. Así con esta modificación el motor del servo gira sin parar cuando

se aplique una señal PWM. Como última modificación al servo, se abrieron los engranajes y se observó que una de las ruedas tiene un pin que se usa para restringir la rotación del servomotor a solo 180 grados, se retira ese pin para que pueda girar libremente 360 grados. Esto se puede evidenciar en la Figura 9.

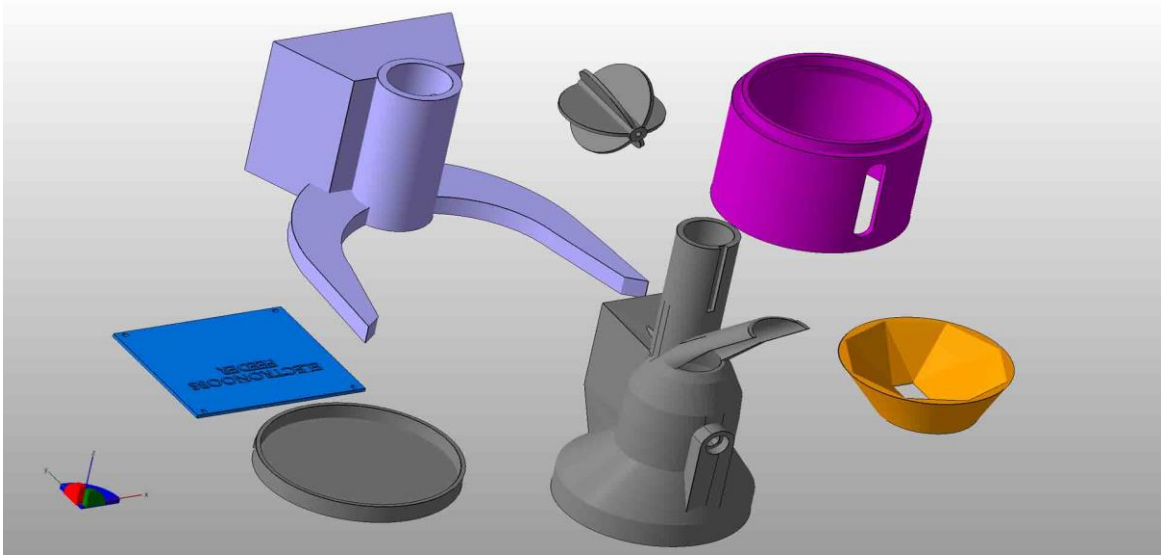

Figura 8. Piezas 3D del dispensador
Fuente: extraído de http://electronoobs.com/eng_arduino_tut163_stl1.php

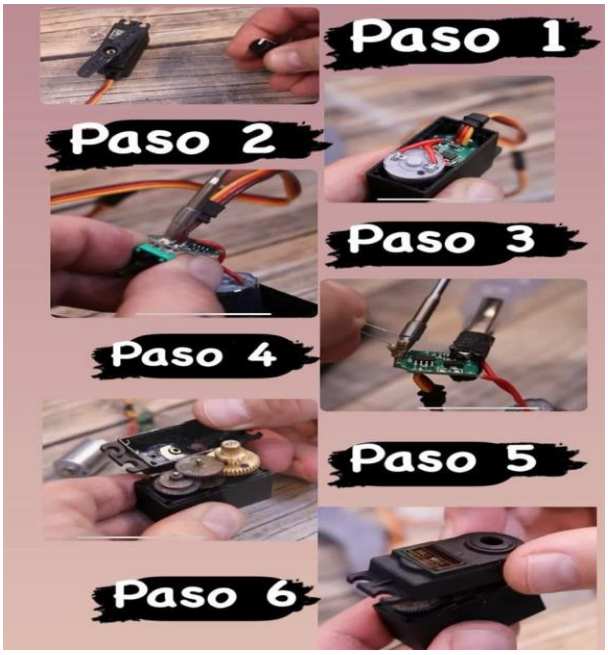

Figura 9. Configuración del servo MG995
Fuente: diseño propio

Adicionalmente, se desarrolló un código en un microcontrolador para permitir la dispensación de comida en porciones pequeñas o grandes, así como la programación de intervalos de tiempo específicos para activar la liberación de comida cuando se detecta la presencia de un animal. Se inició este proceso implementando un código que permitió dosificar alimento a través de un servo y una ESP32 CAM con reconocimiento facial usando el sistema FACEID ya que es para una sola mascota de casa.

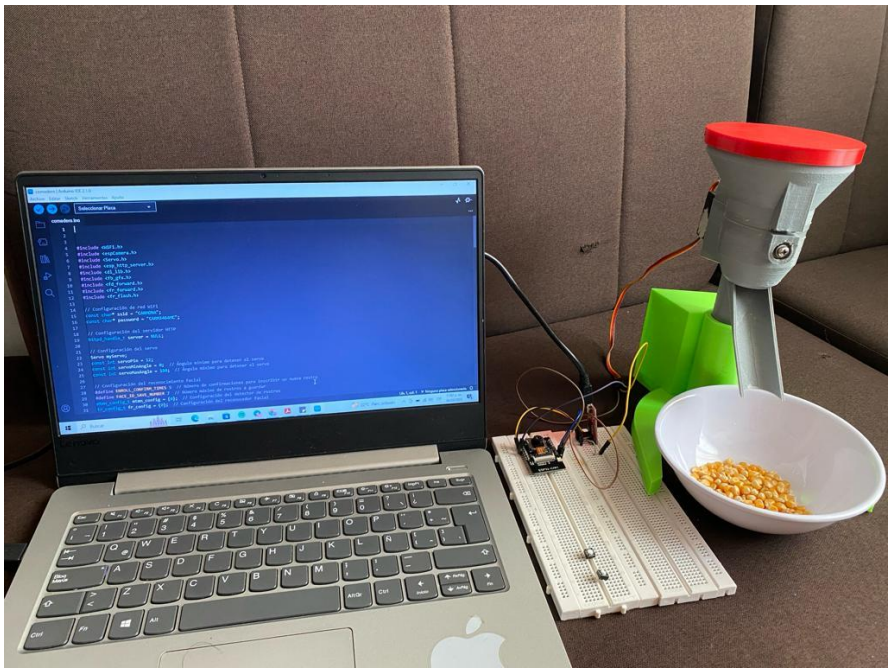

Figura 10. Programación de funciones para realizar dispensación personalizada
Fuente: diseño propio

El código se muestra en el **Anexo A**. Este código controla un servomotor y permite girarlo de 0 a 180 grados con dos botones y detenerlo con un interruptor ON/OFF. El uso de resistencias pull-up internas en los pines de entrada asegura que los estados sean bajo cuando se activan los botones o se enciende el interruptor y permite accionar el servo y realizar la dosificación.

6.2 Entrenamiento de la red neuronal

Este proceso permitió obtener un sistema de reconocimiento facial basado en una Red Neuronal que permita la identificación precisa de los animales que se acercan al alimentador,

además de calcular y dispensar la cantidad adecuada de comida según el tamaño y las necesidades nutricionales específicas de cada animal.

Se comenzó con la creación de un banco de imágenes, adecuando etiquetas nombradas como Gato1 y Gato2. Por cada gato se grabaron 5 videos de 5 minutos de duración. Luego, cada video se dividió en fragmentos utilizando un código, el cual se detalla en el **Anexo B**, ejecutado en Google Colab y en lenguaje Python que permitió obtener un conjunto de 11.958 imágenes para Gato1 y 11.404 imágenes para Gato2, las cuales son utilizadas en el entrenamiento.

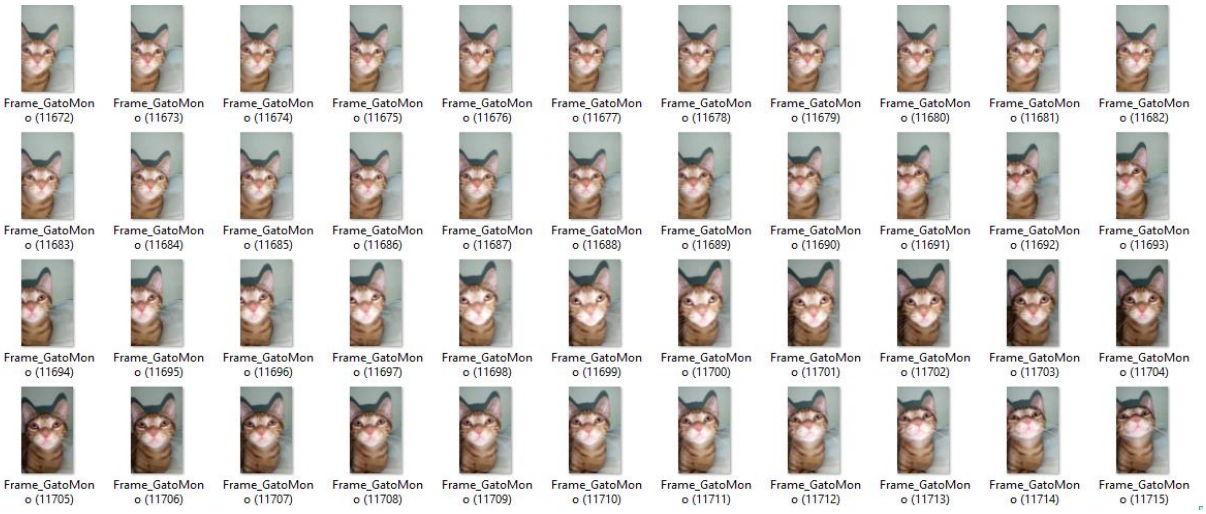

Figura 11. Imágenes del Gato 1 utilizadas en el entrenamiento de la red neuronal
Fuente: diseño propio

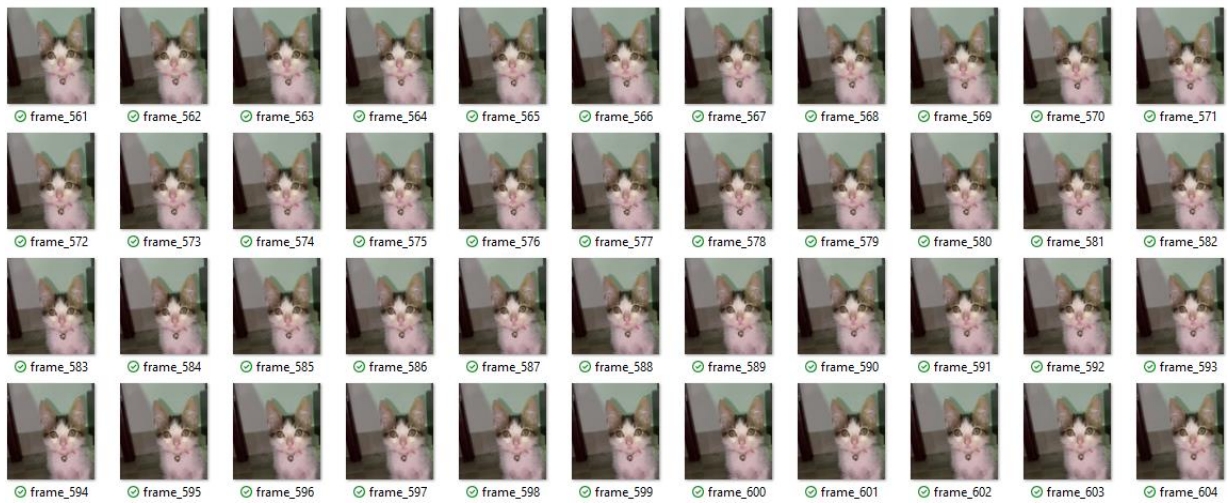

Figura 12. Imágenes del Gato 2 utilizadas en el entrenamiento de la red neuronal
Fuente: diseño propio

Una vez adecuado el banco de imágenes, se procedió con el entrenamiento haciendo uso del código mostrado en el **Anexo C**. Este código inicia con la construcción de un conjunto de datos para entrenar y probar modelos de aprendizaje automático. El código toma un conjunto de imágenes originales y las divide en conjuntos de entrenamiento, validación y prueba. Adicionalmente, mediante el código detallado en el **Anexo D** se configuran algunas rutas y parámetros relacionados con el procesamiento de un conjunto de datos de imágenes de gatos.

Los parámetros requeridos para llevar a cabo el proceso de entrenamiento se sintonizaron así:

- Tasa de aprendizaje inicial: 0.0001
- Tamaño de lote: 64
- Número de épocas: 12

Se hizo uso de la arquitectura de red neuronal EfficientNet B4 en el proceso de entrenamiento. La figura 13 muestra la evolución del valor por época de la función de pérdida (loss) y precisión (acc).

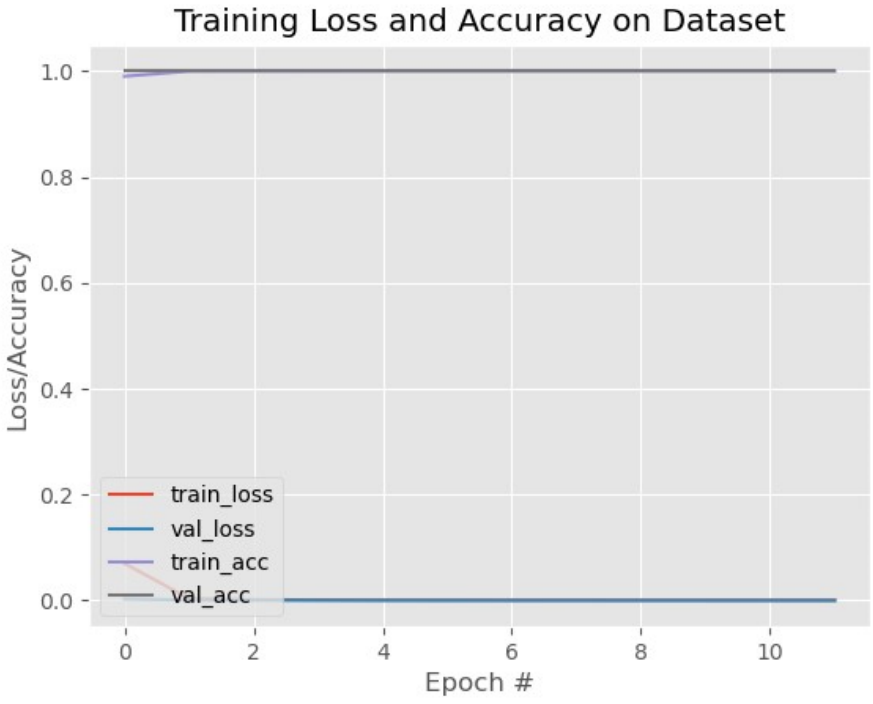

Figura 13. Resultado del entrenamiento de la red neuronal
Fuente: diseño propio

6.3 Desempeño del sistema de reconocimiento

Una vez que el proceso de entrenamiento está completo, el siguiente paso es la programación en el entorno de Arduino IDE descrito en el **Anexo E**. La función principal de este código consiste en la detección de rostros en imágenes capturadas y, al identificar ciertos rostros, se activa un servomotor. Esta funcionalidad es esencial para asegurar que únicamente la mascota designada tenga acceso al dispensador, lo que tiene un impacto significativo en la seguridad y el bienestar de las mascotas. El código permite una alimentación personalizada y garantiza que solo la mascota destinataria reciba la comida.

Luego se procede con la realización de pruebas para evaluar el proceso de detección facial entre el Gato1 y el Gato2. Durante estas pruebas se utilizó un conjunto de 5 imágenes por cada gato ejecutando el modelo de la red neuronal entrenada haciendo uso de un equipo de cómputo con procesador Intel(R) Core(TM) i5-9500 CPU @ 3.00GHz, 16 GB de RAM y sistema operativo Windows 10. Cada imagen fue procesada individualmente para medir el tiempo que la red neuronal tarda en realizar la detección facial, y se registró el porcentaje de precisión.

Resulta evidente que el desempeño del sistema de reconocimiento se vio comprometido debido al empleo de fragmentos durante el entrenamiento que contenían tanto el rostro como parte del cuerpo de cada gato. Al evaluar el sistema con imágenes que solo muestran el rostro se experimentó una disminución en la precisión y en algunas situaciones identificación errónea. El sistema provee una mejor precisión con imágenes de prueba que incluyen tanto el rostro como el cuerpo de cada gato, tal como se detalla en la Tabla 1.

Tabla 1.
Desempeño del modelo entrenado

Imagen de entrada	**Tiempo de detección (s)**	**Resultado (%)**
Gato 1A 	3.8256	Gato1 99.96

Tabla 1.
(Continuación)

Imagen de entrada	Tiempo de detección (s)	Resultado (%)
Gato 1B 	3.7698	Gato1 99.89
Gato 1C 	3.8549	Gato1 79.12
Gato 1D 	3.7494	Gato1 63.51
Gato 1E 	4.0700	Gato1 57.44
Gato 2A 	3.8247	Gato2 99.93
Gato 2B 	3.9000	Gato2 99.99
Gato 2C 	4.0332	Gato2 99.98

Tabla 1.
(Continuación)

Imagen de entrada	Tiempo de detección (s)	Resultado (%)
Gato 2D 	3.7754	Gato1 88.28
Gato 2E 	4.3360	Gato2 99.99

Fuente: diseño propio

Finalmente, durante el proceso de implementación del código para la detección facial en ESP32CAM mediante TensorFlow Lite, se logró avanzar hasta la fase de intentos de pruebas con uno de los modelos entrenado. Sin embargo, la ejecución de las pruebas se vio frustrada por desafíos significativos relacionados con bibliotecas y compatibilidad, que impactaron directamente la capacidad de realizar la detección facial en condiciones óptimas. El inconveniente clave surgió con la biblioteca “Servo” inicialmente seleccionada, revelando su incompatibilidad con el ESP32. La solución inmediata consistió en cambiar a la biblioteca “ESP32Servo” para lograr una integración exitosa y asegurar el control adecuado del servomotor.

Estos desafíos, aunque impidieron las pruebas inmediatas, han proporcionado valiosas lecciones y sentado las bases para futuros refinamientos, consolidando la experiencia de desarrollo en entornos más complejos y señalando áreas específicas para mejoras continuas. Lamentablemente, a pesar de los esfuerzos, se ha podido encontrar una solución satisfactoria para el error específico que se ha presentado durante la ejecución del código en ESP32CAM. El mensaje de error, que incluye referencias a "esp_core_dump_partition" y "esp_core_dump_flash", indica la ausencia de una partición de volcado de núcleo, lo que ha impedido la realización exitosa de pruebas. Este inconveniente podría ser indicativo de un problema más profundo relacionado con la configuración del ESP32 o la incompatibilidad con el entorno de desarrollo actual. En este punto, se deben explorar alternativas y evaluar

detenidamente las configuraciones que podrían ser claves para superar este obstáculo y permitir la continuación de las pruebas de detección facial.

7. Conclusiones

Fue posible desarrollar un dispensador automatizado de bajo costo para mascotas con el propósito de mejorar la calidad de vida de los animales domésticos y simplificar la rutina de los dueños que enfrentan limitaciones de tiempo, al mismo tiempo que se promueve el bienestar y salud de las mascotas.

Este proyecto representa un avance sustancial en la mejora del bienestar de las mascotas a través de la implementación de un dispensador automatizado accesible. El diseño eficiente y económico del hardware, junto con la integración de tecnología de reconocimiento facial, garantiza no solo la comodidad en la alimentación de las mascotas, sino también su seguridad y autenticación.

Las pruebas realizadas han confirmado un desempeño aceptable del sistema de reconocimiento. Sin embargo, para elevar aún más su eficiencia y capacidad de adaptación, se propone un plan de mejora integral. En primer lugar, la implementación de técnicas avanzadas de aumento de datos, que incluyen variaciones en la iluminación, fondos y ángulos de captura, permitirá que el modelo se adapte de manera más efectiva a condiciones adversas y mejore su capacidad de detección.

Además, se sugiere explorar la incorporación de capas adicionales en el modelo, específicamente diseñadas para procesar información contextual sobre el cuerpo del gato. Esta mejora tiene el potencial de fortalecer la precisión del reconocimiento en diversas situaciones, abordando así las limitaciones identificadas en el proyecto.

En conjunto, estas mejoras tienen como objetivo superar las limitaciones actuales y potenciar la eficiencia del dispensador automatizado, contribuyendo a una convivencia más saludable y compasiva entre los dueños y sus mascotas. La implementación de estas sugerencias consolidará la posición de este proyecto como una solución innovadora y efectiva para el cuidado de animales domésticos.

8. Recomendaciones

En el contexto de este proyecto, se recomienda la colaboración y consulta continua con expertos en bienestar animal y etología para garantizar que las funciones de dispensación y el diseño del sistema sean compatibles con las necesidades nutricionales y conductuales de las mascotas. Además, se sugiere llevar a cabo pruebas exhaustivas no solo en entornos controlados, sino también en situaciones del mundo real para evaluar la durabilidad y la eficacia del dispensador en diversas circunstancias. Se insta a la consideración de medidas de seguridad adicionales para prevenir posibles accidentes y garantizar la seguridad de las mascotas y su bienestar mientras utilizan el dispensador automatizado. La promoción activa del uso responsable de la tecnología entre los dueños de mascotas, también se presentan como recomendaciones clave para maximizar el impacto positivo de esta innovación en la comunidad.

9. Referencias bibliográficas

Barbieri, A. (11 de Noviembre de 2018). *El reconocimiento facial en animales, un modelo de «vigilancia» beneficioso*. Recuperado el 03 de Noviembre de 2023, de https://nobbot.com/reconocimiento-facial-animales/

Dassault Systèmes. (2023). *Impresión 3D*. Recuperado el 05 de Octubre de 2023, de https://www.3ds.com/es/make/guide/process/3d-printing

Electronoobs. (14 de Noviembre de 2021). *Cat feeder - 3D Files*. Recuperado el 08 de Agosto de 2023, de http://electronoobs.com/eng_arduino_tut163_stl1.php

Inteligencia Futura. (07 de Abril de 2020). *Redes neuronales profundas, o Perceptron contra los perros y gatos*. Recuperado el 28 de Octubre de 2023, de https://inteligenciafutura.mx/blog/redes-neuronales-profundas-o-perceptron-contra-los-perros-y-gatos

Kaspersky. (2023). Recuperado el 20 de Septiembre de 2023, de Reconocimiento facial: definición y explicación: https://latam.kaspersky.com/resource-center/definitions/what-is-facial-recognition

Noticias HSB. (16 de Septiembre de 2022). *Preocupante cifra de mascotas abandonadas en Colombia*. Recuperado el 24 de Agosto de 2023, de https://www.hsbnoticias.com/preocupante-cifra-de-mascotas-abandonadas-en-colombia/

Pascual, C. (18 de Enero de 2022). *ESP32 CAM introducción y primeros pasos*. Recuperado el 20 de Octubre de 2023, de https://programarfacil.com/esp32/esp32-cam/

Published. (07 de Junio de 2016). *Cómo elegir la nutrición para gatos correcta para tu gato*. Recuperado el 12 de Septiembre de 2023, de Hill's Pet Nutrition: https://www.hillspet.com.mx/cat-care/nutrition-feeding/choosing-the-right-cat-nutrition

10. Bibliografía

P. (2023, 26 de octubre). *Elegir el mejor alimento para gatitos: Qué buscar, evitar*. Nutrición para mascotas de Hill; Editor. https://www.hillspet.com.mx/cat-care/nutrition-feeding/elegir-el-mejor-comida-para-gatitos

C1F3R. (2021, 30 de mayo). *Sistema de detección de objetos con ESP32-CAM y tensorflow* . CiferTech; C1F3R. https://cifertech.net/sistema-de-deteccion-de-objetos-con-esp32-cam-y-tensorflow/

García, ER (2018, 10 de julio). *La impresión 3D en el punto de mira de la UE por la fabricación de copias* . El Español. https://www.elespanol.com/omicrono/tecnologia/20180710/impresion-punto-mira-ue-fabricacion-copias/321469170_0.html

¡Increíble! Transmite videos en ESP32 con reconocimiento facial. (2023, mayo 31). *Top Electric*. https://www.topelectric.es/como-usar-el-modulo-de-camara-esp32-para-transmision-de-video-y-reconocimiento-facial/

¿Qué son las redes neuronales? (s/f). Ibm.com. Recuperado el 11 de noviembre de 2023, de https://www.ibm.com/es-es/topics/neural-networks

11. Anexos

Anexo A. Código de control del servomotor

```
#include <Servo>
```

1. Se incluye la biblioteca **Servo**, que se utiliza para controlar el servo motor.

```
Servo miServo;
```

2. Se crea un objeto llamado **miServo** de la clase **Servo**, que se utilizará para controlar el servo.

```
int servoPin = 9; int boton1Pin = 10; int boton2Pin = 11; int switchPin = 12;
```

3. Se declaran variables para almacenar los números de pin a los que están conectados el servo (**servoPin**), el primer botón (**boton1Pin**), el segundo botón (**boton2Pin**) y el interruptor ON/OFF (**switchPin**).

```
int estadoBoton1 = 0; int estadoBoton2 = 0; int estadoSwitch = 0;
```

4. Se declaran variables para almacenar el estado de los botones y el interruptor. Inicialmente se establecen en 0.

```
void setup() { miServo.attach(servoPin); pinMode(boton1Pin, INPUT_PULLUP); pinMode(boton2Pin, INPUT_PULLUP); pinMode(switchPin, INPUT_PULLUP); }
```

5. En la función **setup()**, se realiza la configuración inicial:
   - **miServo.attach(servoPin)**: Se inicia el objeto **miServo** y se le asigna el pin donde está conectado el servo.
   - **pinMode(boton1Pin, INPUT_PULLUP)**: Se configura el pin del botón 1 como entrada con resistencia pull-up interna.
   - **pinMode(boton2Pin, INPUT_PULLUP)**: Se configura el pin del botón 2 de la misma manera.
   - **pinMode(switchPin, INPUT_PULLUP)**: Se configura el pin del interruptor de la misma manera.

```
void loop() { estadoBoton1 = digitalRead(boton1Pin); estadoBoton2 = digitalRead(boton2Pin); estadoSwitch = digitalRead(switchPin);
```

6. En la función **loop()**, se inicia el bucle principal que se ejecuta continuamente. Se leen los estados de los botones y el interruptor y se almacenan en las variables correspondientes.

```
if (estadoSwitch == LOW) {
```

7. Se verifica si el interruptor está en la posición encendida (LOW).

```
if (estadoBoton1 == LOW) { miServo.write(0); } else if (estadoBoton2 == LOW) { miServo.write(180); } else { miServo.write(90); } } else { miServo.write(90); }
```

8. Si el interruptor está encendido, se comprueba el estado de los botones:
   - Si el botón 1 está presionado (LOW), el servo se mueve a 0 grados.
   - Si el botón 2 está presionado (LOW), el servo se mueve a 180 grados.

- Si ningún botón está presionado, el servo se detiene en 90 grados.

9. Si el interruptor está apagado, el servo se detiene en 90 grados, independientemente de la presión de los botones

Anexo B. Código de fragmentación de videos

1. Importación de bibliotecas:

pythonCopy code
import cv2 import os from google.colab import files

- **cv2**: Importa la biblioteca OpenCV, que se utiliza para trabajar con imágenes y videos.
- **os**: Importa la biblioteca para operaciones relacionadas con el sistema operativo, como crear directorios y obtener la carpeta actual.
- **files**: Importa la biblioteca de Google Colab para cargar y descargar archivos desde y hacia el entorno Colab.

2. Carga de videos:

pythonCopy code
uploaded = files.upload()

Esta línea permite al usuario cargar videos desde su computadora al entorno de Google Colab. Los videos cargados se almacenan en la variable **uploaded**.

3. Obtención de la carpeta actual:

pythonCopy code
current_folder = os.getcwd()

Obtiene la ruta de la carpeta actual en Google Colab y la almacena en la variable **current_folder**.

4. Definición de nombres de videos:

pythonCopy code
video_names = list(uploaded.keys())

Crea una lista de nombres de los videos cargados, tomando las claves del diccionario **uploaded**.

5. Definición de tiempos de inicio y fin:

pythonCopy code
start_times = [0, 5, 10, 15, 20, 25, 60] # Tiempos de inicio en segundos end_times = [5, 10, 20, 40, 50, 60, 70] # Tiempos de fin en segundos

Define dos listas: **start_times** (tiempos de inicio en segundos) y **end_times** (tiempos de fin en segundos) que especifican los intervalos de tiempo en los que se extraerán los fragmentos de video.

6. Creación de una carpeta para imágenes:

pythonCopy code
image_folder = "fragmentos_de_imagen" os.makedirs(image_folder, exist_ok=True)

Crea una carpeta llamada "fragmentos_de_imagen" para almacenar las imágenes extraídas. La opción **exist_ok=True** evita errores si la carpeta ya existe.

7. Iteración sobre los videos y extracción de fragmentos de imágenes:

pythonCopy code
for i, video_name in enumerate(video_names):

Este bucle **for** itera a través de los videos cargados y utiliza la función **enumerate** para obtener tanto el índice **i** como el nombre del video **video_name**.

pythonCopy code

	start_time = start_times[i] end_time = end_times[i]
	Obtiene los tiempos de inicio y fin correspondientes al video actual.
	pythonCopy code cap = cv2.VideoCapture(video_name) frame_rate = int(cap.get(5)) # Obtén la tasa de fotogramas del video
	Abre el video utilizando OpenCV y obtiene su tasa de fotogramas.
	pythonCopy code output_folder = os.path.join(image_folder, f"fragment_{i}") os.makedirs(output_folder, exist_ok=True)
	Crea una carpeta de salida específica para las imágenes extraídas del video actual.
	pythonCopy code frame_number = 0 while True: ret, frame = cap.read() if not ret: break
	Inicia un bucle **while** para leer los fotogramas del video actual. La variable **ret** indica si se ha leído correctamente un fotograma.
	pythonCopy code if frame_number >= start_time * frame_rate and frame_number <= end_time * frame_rate: image_name = os.path.join(output_folder, f"frame_{frame_number}.jpg") cv2.imwrite(image_name, frame)
	Comprueba si el número de fotograma actual está dentro del intervalo de tiempo especificado y, en caso afirmativo, guarda ese fotograma como una imagen en la carpeta de salida.
	pythonCopy code frame_number += 1
	Incrementa el número de fotograma.
	pythonCopy code cap.release()
	Libera el objeto de captura de video una vez que se haya terminado de procesar el video actual.
8.	Compresión de las carpetas de fragmentos de imagen en archivos ZIP:
	pythonCopy code import shutil shutil.make_archive(image_folder, 'zip', image_folder)
	Utiliza la biblioteca **shutil** para comprimir la carpeta "fragmentos_de_imagen" en un archivo ZIP.
9.	Descarga de los archivos ZIP:
	pythonCopy code files.download(image_folder + '.zip')
	Descarga el archivo ZIP resultante al sistema local del usuario desde Google Colab.

Anexo C. Código de entrenamiento de red neuronal

```
# USAGE

# python build_dataset.py



# import the necessary packages

from config_file import config_1

from imutils import paths

import random

import shutil

import os
```

Listado y aleatorización de rutas de imágenes: El script recopila las rutas de todas las imágenes en un directorio original (configurado en config_1.ORIG_INPUT_DATASET). Luego, se mezclan las rutas de las imágenes de manera aleatoria.

```


# grab the paths to all input images in the original input directory

# and shuffle them

imagePaths = list(paths.list_images(config_1.ORIG_INPUT_DATASET))

random.seed(42)

random.shuffle(imagePaths)
```

Partición de datos de entrenamiento para validación: Se calcula un nuevo índice i para dividir el conjunto de entrenamiento en entrenamiento y validación, según el valor de config_1.VAL_SPLIT. Esto crea un conjunto de datos de validación separado.

```
# compute the training and testing split

i = int(len(imagePaths) * config_1.TRAIN_SPLIT)

trainPaths = imagePaths[:i]
```

```
testPaths = imagePaths[i:]


# we'll be using part of the training data for validation

i = int(len(trainPaths) * config_1.VAL_SPLIT)

valPaths = trainPaths[:i]

trainPaths = trainPaths[i:]
```

Definición de los conjuntos de datos: Los conjuntos de datos se definen como una lista de tuplas, donde cada tupla contiene el nombre del conjunto ("training", "validation" o "testing"), las rutas de las imágenes y la ubicación de salida (directorio donde se guardarán los datos).

```
# define the datasets that we'll be building

datasets = [

        ("training", trainPaths, config_1.TRAIN_PATH),

        ("validation", valPaths, config_1.VAL_PATH),

        ("testing", testPaths, config_1.TEST_PATH)

]


# loop over the datasets

for (dType, imagePaths, baseOutput) in datasets:

        # show which data split we are creating

        print("[INFO] building '{}' split".format(dType))


        # if the output base output directory does not exist, create it

        if not os.path.exists(baseOutput):

                print("[INFO] 'creating {}' directory".format(baseOutput))
```

```
        os.makedirs(baseOutput)
```

Bucle principal para crear los conjuntos de datos: Se inicia un bucle que recorre los conjuntos de datos definidos. Para cada conjunto de datos, se realizan las siguientes acciones:

Se muestra un mensaje que indica qué conjunto de datos se está creando.

```
    # loop over the input image paths
    for inputPath in imagePaths:
        # extract the filename of the input image along with its
        # corresponding class label
        filename = inputPath.split(os.path.sep)[-1]
        label = inputPath.split(os.path.sep)[-2]

        # build the path to the label directory
        labelPath = os.path.sep.join([baseOutput, label])

        # if the label output directory does not exist, create it
        if not os.path.exists(labelPath):
            print("[INFO] 'creating {}' directory".format(labelPath))
            os.makedirs(labelPath)

        # construct the path to the destination image and then copy
        # the image itself
        p = os.path.sep.join([labelPath, filename])
```

```
            shutil.copy2(inputPath, p)
```

Si el directorio de salida base (baseOutput) no existe, se crea.

Se recorren las rutas de las imágenes en el conjunto actual.

Para cada imagen, se extrae el nombre de archivo y la etiqueta de clase a partir de la ruta.

Se crea la ruta al directorio de salida de la etiqueta.

Si el directorio de salida de la etiqueta no existe, se crea.

La imagen se copia desde su ubicación original a la ubicación de salida, manteniendo el nombre de archivo.

Este script toma un conjunto de imágenes originales y crea tres conjuntos de datos: entrenamiento, validación y prueba. Las imágenes se copian en directorios separados según sus etiquetas de clase. Esto es útil para preparar datos para el entrenamiento y la evaluación de modelos de aprendizaje automático.

Anexo D. Código para configurar algunas rutas

1. **import os**: Importa el módulo 'os' que proporciona funciones para interactuar con el sistema operativo, como trabajar con rutas de archivos y directorios.
2. **ORIG_INPUT_DATASET = "cat_dataset"**: Establece la ruta al directorio original que contiene las imágenes del conjunto de datos de gatos. Puedes cambiar esta ruta al directorio que contenga tus datos.
3. **BASE_PATH = "cat_dataset_t_1"**: Establece la ruta base para el nuevo directorio que contendrá las imágenes después de dividirlas en conjuntos de entrenamiento, validación y prueba. Este nuevo directorio se usará para organizar las imágenes procesadas.
4. **TRAIN_PATH = os.path.sep.join([BASE_PATH, "training"])**: Crea la ruta para el directorio de entrenamiento concatenando **BASE_PATH** con "training". Esto se usará para almacenar las imágenes de entrenamiento.
5. **VAL_PATH = os.path.sep.join([BASE_PATH, "validation"])**: Crea la ruta para el directorio de validación concatenando **BASE_PATH** con "validation". Las imágenes de validación se guardarán aquí.
6. **TEST_PATH = os.path.sep.join([BASE_PATH, "testing"])**: Crea la ruta para el directorio de prueba concatenando **BASE_PATH** con "testing". Las imágenes de prueba se guardarán en este directorio.
7. **TRAIN_SPLIT = 0.75**: Define la fracción del conjunto de datos que se utilizará para entrenamiento. En este caso, se usará el 75% de las imágenes para entrenar.
8. **VAL_SPLIT = 0.1**: Define la fracción del conjunto de entrenamiento que se utilizará para validación. En este caso, el 10% de las imágenes de entrenamiento se utilizarán para validación.
9. **CLASSES = ["cat_001", "cat_002"]**: Enumera los nombres de las clases presentes en el conjunto de datos. En este caso, hay dos clases etiquetadas como "cat_001" y "cat_002". Debes cambiar esto según las clases en tu conjunto de datos.
10. **INIT_LR = 1e-4**: Establece la tasa de aprendizaje inicial para un modelo de entrenamiento. Esta es una hiperparámetro que se utiliza en algoritmos de aprendizaje automático.
11. **BS = 64**: Define el tamaño del lote (batch size) que se utilizará durante el entrenamiento del modelo. Un tamaño de lote de 64 significa que se procesarán 64 imágenes a la vez durante cada iteración de entrenamiento.
12. **NUM_EPOCHS = 10**: Establece el número de épocas (iteraciones completas a través del conjunto de entrenamiento) que se utilizarán para entrenar el modelo.
13. **MODEL_PATH = "cat_1.h5"**: Define la ruta donde se guardará el modelo entrenado en formato HDF5. Esto será el archivo del modelo después de completar el entrenamiento.

Este código se encarga de configurar rutas y parámetros clave para el procesamiento y entrenamiento de un modelo de aprendizaje automático en un conjunto de datos de imágenes de gatos. Los valores de los parámetros y rutas se pueden personalizar según tus necesidades y el conjunto de datos específico que estás utilizando.

Anexo E. Código Arduino Tensor Flow usando ESP32

El código comienza incluyendo las bibliotecas necesarias, como Servo para el control del servo y otras bibliotecas relacionadas con la cámara y TensorFlow Lite.

```
#include <Arduino.h>
#include "esp_camera.h"
#include <TensorFlowLite.h>
#include "image_provider.h"
#include <Servo.h>  // Incluye la librería para controlar el servo

// Incluye el modelo .tflite generado para la detección facial
#include "modelo_facenet_tflite.h"

// Crea un objeto de TensorFlowLite para manejar el modelo
TfLiteModel* modelo = TfLiteModelCreate(modelo_facenet_tflite);

// Define un objeto para el intérprete de TensorFlow Lite
TfLiteInterpreter* intérprete = TfLiteInterpreterCreate(modelo);
```

•Luego, se crea un objeto Servo llamado miServo y se define el pin al que está conectado el servo (pinServo).

```
// Define un objeto para el servo
Servo miServo;
const int pinServo = 13;  // Elige el pin GPIO al que está conectado el servo
```

•En la función setup(), se inicia la comunicación serial para la depuración y se inicializa el servo usando el pin especificado.

```
void setup() {
  Serial.begin(115200);

  // Inicializa el servo
  miServo.attach(pinServo);
```

•Se establece la configuración de la cámara ESP32CAM en la sección // ... (configuración de la cámara).

•Se crea un modelo TensorFlow Lite y un intérprete en la sección correspondiente

```
  // Inicializa la cámara ESP32CAM
```

```
  camera_config_t config;
  // ... (configuración de la cámara, configuración de pines, resolución, etc.)

  // Inicializa el intérprete de TensorFlow Lite
  // ...

  // Otras inicializaciones
  // ...
}
```

•La función loop() se encarga de capturar una imagen de la cámara, preprocesarla y cargarla en el intérprete de TensorFlow Lite. Luego, se obtienen los resultados de la detección facial a través de la salida del modelo.

```
void loop() {
  // Captura una imagen desde la cámara
  camera_fb_t* frame = esp_camera_fb_get();
  // ...

  // Preprocesa la imagen y cárgala en el intérprete de TensorFlow Lite
  // ...

  // Obtiene la salida del modelo (resultados de la detección facial)
  const TfLiteTensor* salida = TfLiteInterpreterGetOutputTensor(intérprete, 0);
  const float* resultados = salida->data.f;

  // Procesa los resultados de la detección facial
  // ...

  // Si se detecta un "gato 1" o "gato 2", activa el servo
  if (resultados[0] > umbral || resultados[1] > umbral) {
    miServo.write(90);  // Mueve el servo a una posición específica (90 grados, por ejemplo)
  } else {
    miServo.write(0);   // Vuelve el servo a la posición inicial
  }

  // Libera la imagen capturada
  // ...
}
```

•Se procesan los resultados según sea necesario. Si se detecta un "gato 1" o "gato 2" , se activa el servo moviéndolo a 90 grados. En caso contrario, el servo se mueve de nuevo a la posición inicial (0 grados).
•Finalmente, se libera la imagen capturada por la cámara.