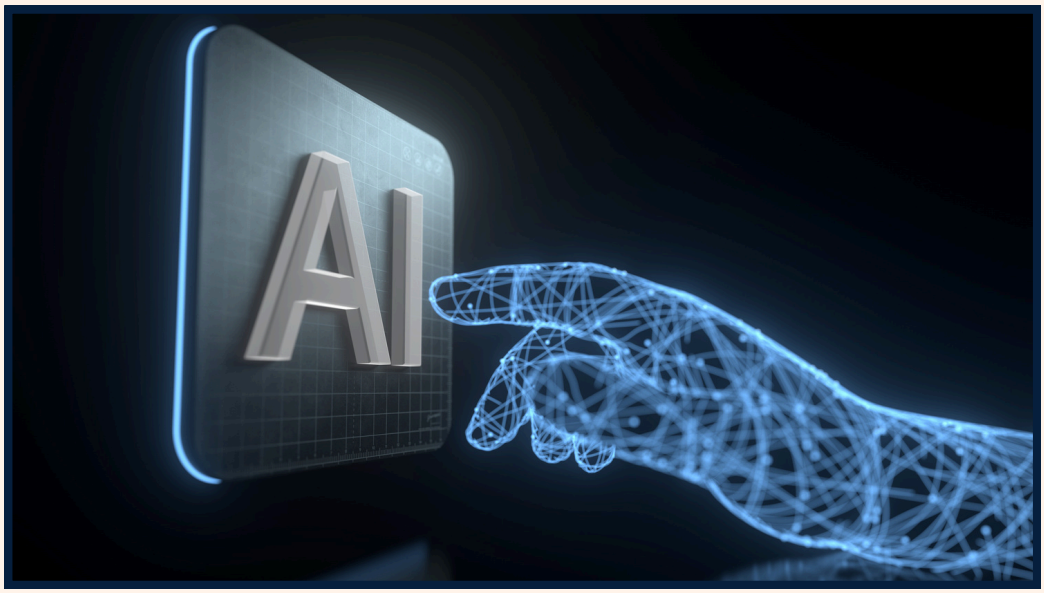
MAYO 2024

Inteligencia Artificial Empotrada

Jose Luis Galán | Alberto Andrés | Andrés Toledo
Gonzalo Albelda | Gonzalo Martín



ÍNDICE

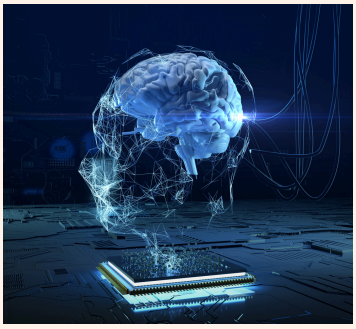
1. Introducción
   - Contexto 3
   - La IA en los microcontroladores 4
2. Módulo de aceleración IA
   - Introducción 4
   - ¿Qué es un módulo de aceleración de IA? 5
   - Funcionamiento del módulo en el ESP32 5
3. La librería ‘esp-tflite-micro'
   - Descripción 6
   - Funcionamiento de la Librería 7
   - Proceso de Uso 7
   - Ventajas 8
4. Descripción del proyecto
   - Estructura general 9
   - Función setup() 10
   - Función loop() 12
   - Diagrama de ficheros 13
5. Anexos
   - Programación de redes neuronales 14
   - Aprendizaje automático 16
   - Operaciones Matemáticas 17
   - Librerías Espressif 18

Introducción

Contexto

Las Inteligencias Artificiales (IA) están diseñadas para simular el pensamiento humano y realizar tareas que normalmente precisarían una implicación humana. Gracias a sus avanzadas capacidades, las IA tienen una amplia variedad de usos, que incluyen el entendimiento tanto de lenguajes de programación como de escritura, la identificación de patrones complejos, la generación de explicaciones detalladas, entre otros. Estas herramientas no solo emulan el proceso cognitivo humano, sino que también pueden superarlo en términos de velocidad y precisión en diversas aplicaciones prácticas.

El uso de la inteligencia artificial (IA) ha crecido exponencialmente en los últimos años por las múltiples ventajas que ofrece. La IA, rama de la informática dedicada a crear sistemas capaces de simular el pensamiento humano y realizar tareas que requerirían intervención humana, ha revolucionado aspectos de la vida cotidiana. Desde asistentes virtuales como Siri y Alexa, que entienden y responden a comandos de voz, hasta aplicaciones de recomendación en plataformas como Netflix y Amazon, la IA se ha integrado en nuestras rutinas diarias.



Además, en el ámbito de la programación, la IA permite traducir código entre diferentes lenguajes, desarrollar funciones automatizadas a partir de simples descripciones de los resultados deseados y analizar y explicar códigos ya existentes. Estas tecnologías abarcan áreas como el aprendizaje, la percepción, el razonamiento, la resolución de problemas, la comprensión del lenguaje natural y la visión por computadora, facilitando y optimizando el trabajo de los programadores y mejorando la eficiencia y la calidad de diversas aplicaciones y servicios diarios.

Hoy en día el uso de estas se ha vuelto recurrente, ya que se usan para casi cualquier ámbito. La inteligencia artificial no solo ha transformado la programación, sino que también ha encontrado aplicaciones en una variedad de sectores como la medicina, la educación, el transporte, y el entretenimiento, entre otros. Su capacidad para manejar grandes volúmenes de datos y ejecutar tareas complejas con precisión ha convertido a la IA en una herramienta indispensable en el mundo moderno. A medida que la tecnología sigue avanzando, es probable que veamos aún más innovaciones y aplicaciones de la IA que continuarán redefiniendo nuestras capacidades y posibilidades en diversos campos.

La IA en los microcontroladores

A medida que la inteligencia artificial sigue integrándose en diversos aspectos de nuestra vida diaria, su influencia se extiende también al campo de la tecnología de hardware, específicamente en los microcontroladores. Estos pequeños dispositivos, fundamentales para operar sistemas electrónicos, comienzan a incorporar capacidades de IA para mejorar su rendimiento y funcionalidad.

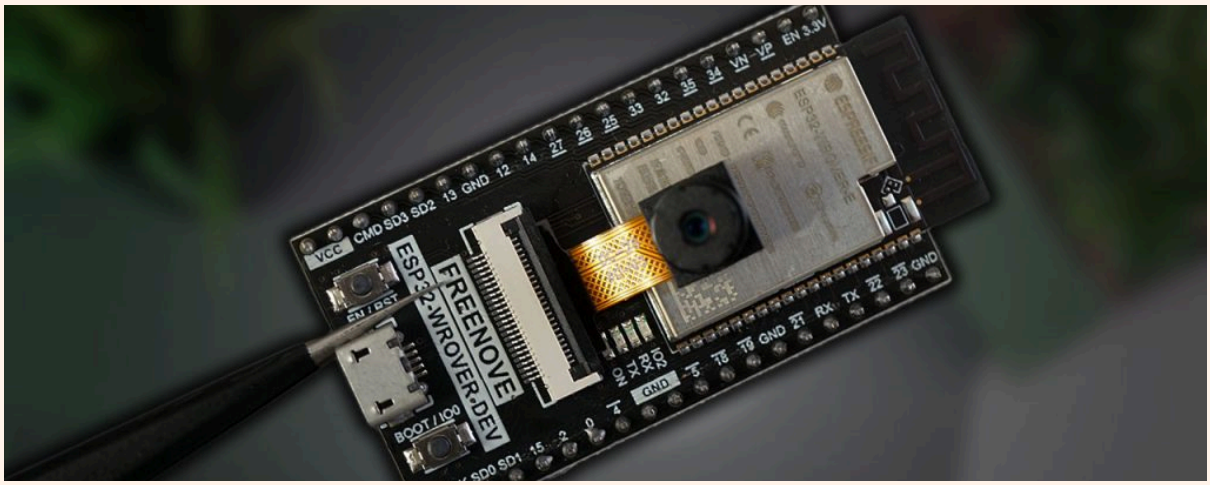
La combinación de IA y microcontroladores abre un abanico de posibilidades, desde la automatización avanzada hasta la creación de sistemas inteligentes más eficientes y autónomos. Un ejemplo destacado de esta integración es el microcontrolador ESP32S3, que ofrece numerosas ventajas gracias a sus capacidades mejoradas de procesamiento y conectividad.

En el siguiente apartado, exploraremos cómo la IA se está implementando en los microcontroladores, centrándonos en las ventajas específicas que aporta el ESP32S3.

Módulo de aceleración IA

Introducción

Espressif Systems se ha implicado en el desarrollo dentro del campo de la inteligencia artificial, la ESP32 es una serie de microcontroladores de bajo costo y alto rendimiento que han ido desarrollando. Estos microcontroladores son populares en aplicaciones de Internet de las Cosas (IoT) debido a su capacidad de conectividad Wi-Fi y Bluetooth. Algunas variantes de ESP32 están equipadas con un módulo de aceleración de inteligencia artificial (IA), también conocido como acelerador de IA. Este módulo permite realizar tareas de procesamiento de IA de manera más eficiente que usando solo la CPU principal.

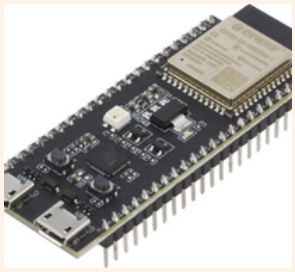


¿Qué es un módulo de aceleración de IA?

Un módulo de aceleración de IA es un hardware especializado diseñado para realizar operaciones de IA, como inferencias de redes neuronales, más rápidamente y con mayor eficiencia energética que una CPU o una GPU general. Estos módulos pueden estar optimizados para realizar operaciones matemáticas comunes en algoritmos de IA, como multiplicaciones y sumas matriciales.

Funcionamiento del módulo en el ESP32

Un módulo de aceleración de IA es un hardware especializado diseñado para realizar operaciones de IA, como inferencias de redes neuronales, más rápidamente y con mayor eficiencia energética que una CPU o una GPU general. Estos módulos pueden estar optimizados para realizar operaciones matemáticas comunes en algoritmos de IA, como multiplicaciones y sumas matriciales.



Arquitectura de Hardware

- **CPU y co-procesadores:** La ESP32 puede tener múltiples núcleos de CPU (dos núcleos Xtensa LX7 en la ESP32-S3), junto con co-procesadores especializados.

- **Unidad de procesamiento de IA:** Algunas variantes pueden incluir una unidad específica para acelerar operaciones de IA, como la ESP32-S3 que cuenta con instrucciones vectoriales para acelerar cálculos matriciales, esenciales en tareas de IA.

Optimización de Operaciones Matemáticas

- **Instrucciones Vectoriales:** El acelerador de IA puede ejecutar instrucciones vectoriales que procesan múltiples datos en paralelo, lo cual es fundamental en operaciones de redes neuronales, como las multiplicaciones de matrices.

- **Procesamiento en paralelo**: Esta capacidad permite realizar inferencias de redes neuronales más rápidamente al procesar varias operaciones simultáneamente en lugar de secuencialmente.

*** ANEXO III**

Soporte de Software

- **Bibliotecas y SDKs:** Espressif proporciona bibliotecas y kits de desarrollo de software (SDKs) que facilitan el uso del módulo de IA. Ejemplos incluyen ESP-DSP y ESP-NN.

- **Modelos Preentrenados:** Los desarrolladores pueden usar modelos de IA preentrenados que se ejecutan de manera eficiente en el hardware acelerado de la ESP32.

*** ANEXO IV**

Interacción con la CPU Principal

- **Tareas Descentralizadas:** La CPU principal puede delegar tareas específicas de IA al módulo de aceleración, permitiendo que la CPU se enfoque en otras tareas o reduzca su carga de trabajo.

- **Comunicación Eficiente:** A través de interfaces de alta velocidad, los datos pueden transferirse rápidamente entre la CPU y el módulo de aceleración de IA.

La librería ‘esp-tflite-micro'

Descripción

Desarrollada por Espressif es una adaptación de TensorFlow Lite Micro (TFLM) para los microcontroladores de la serie ESP32, incluyendo el ESP32-S3. TensorFlow Lite Micro es una versión reducida de TensorFlow Lite diseñada específicamente para dispositivos con recursos limitados, como microcontroladores, permitiendo ejecutar modelos de aprendizaje automático en estos dispositivos.

Esta se integra con ESP-IDF (Espressif IoT Development Framework), el entorno de desarrollo utilizado para programar los microcontroladores ESP32. Esta integración facilita el uso de TensorFlow Lite Micro en aplicaciones desarrolladas para los microcontroladores ESP32.

Enlace: https://github.com/espressif/esp-tflite-micro

Funcionamiento de la Librería

1. Integración con TensorFlow Lite Micro: Se basa en TensorFlow Lite Micro, que está optimizado para ejecutarse en entornos con recursos limitados. Esto incluye soporte para operaciones matemáticas y modelos de inferencia necesarios para ejecutar modelos de aprendizaje automático.

2. Compatibilidad con ESP-IDF: Se integra con ESP-IDF (Espressif IoT Development Framework), el entorno de desarrollo utilizado para programar los microcontroladores ESP32. Esta integración facilita el uso de TensorFlow Lite Micro en aplicaciones desarrolladas para los microcontroladores ESP32.

3. Arquitectura de la Librería: Está diseñada para proporcionar una interfaz eficiente y fácil de usar para la carga, configuración y ejecución de modelos de aprendizaje automático en los microcontroladores ESP32. Esta librería incluye componentes clave como el Model Loader, que se encarga de cargar modelos optimizados de TensorFlow Lite en el microcontrolador, el Interpreter, que ejecuta el modelo gestionando la inferencia en tiempo real, y el Ops Resolver, que resuelve y ejecuta las operaciones definidas en el modelo. Además, cuenta con el Micro Interpreter, que maneja la inferencia en un entorno de memoria estática, lo que es ideal para microcontroladores con recursos limitados. Esta arquitectura permite la integración eficiente de modelos de aprendizaje automático en aplicaciones desarrolladas con ESP-IDF, optimizando el uso de memoria y recursos de procesamiento.

4. Proceso de Uso:

Paso 1: Preparar el Modelo:

- Un modelo de TensorFlow se entrena y optimiza utilizando TensorFlow o TensorFlow Lite.
- El modelo se convierte a un formato compatible con TensorFlow Lite Micro (típicamente un archivo .tflite).

Paso 2: Integrar el Modelo en el Proyecto:

- El modelo .tflite se incluye en el proyecto ESP-IDF.
- Se escribe código en C/C++ para cargar el modelo utilizando la librería esp-tflite-micro.

Paso 3: Configurar e Inicializar:

- Se inicializan los componentes necesarios de la librería, como el intérprete y los tensores.
- Se configura el entorno de ejecución, asignando memoria estática para los tensores y operaciones.

Paso 4: Ejecución de Inferencia:

- Se cargan los datos de entrada al modelo.
- Se ejecuta el intérprete para obtener los resultados de inferencia.
- Los resultados se procesan o se utilizan según la aplicación específica.

Ventajas

1. Optimización para recursos limitados: La librería esp-tflite-micro está especialmente diseñada para ejecutarse en microcontroladores como los ESP32, que tienen recursos limitados en comparación con los sistemas más potentes. Esto significa que puede funcionar eficientemente con memoria y capacidad de procesamiento restringidas. Los microcontroladores suelen tener apenas unos pocos kilobytes de RAM y almacenamiento, por lo que la librería está optimizada para usar una memoria estática y minimizar el uso de recursos. Esto permite que dispositivos pequeños y económicos puedan ejecutar modelos de inteligencia artificial, haciendo posible la implementación de funciones avanzadas de IA en aplicaciones donde antes era inviable debido a las limitaciones de hardware.

2. Fácil integración: esp-tflite-micro se integra sin problemas con ESP-IDF (Espressif IoT Development Framework), el entorno de desarrollo utilizado para programar los microcontroladores ESP32. Esta integración simplifica el proceso de desarrollo, ya que los desarrolladores pueden utilizar las herramientas y bibliotecas familiares de ESP-IDF junto con las capacidades de IA de TensorFlow Lite Micro. Esto reduce la curva de aprendizaje y permite a los desarrolladores enfocarse en la creación de aplicaciones innovadoras sin preocuparse por los detalles de la integración de la IA. Además, Espressif proporciona documentación y ejemplos claros, lo que facilita aún más la incorporación de modelos de IA en proyectos IoT.

3. Flexibilidad: La librería soporta una amplia gama de modelos y operaciones de TensorFlow Lite, permitiendo la ejecución de diversos tipos de modelos de aprendizaje automático, desde redes neuronales simples hasta arquitecturas más complejas. Esto significa que los desarrolladores no están limitados a un tipo específico de modelo o aplicación. La capacidad de manejar diferentes tipos de modelos y operaciones hace que esp-tflite-micro sea adaptable a una variedad de casos de uso, incluyendo reconocimiento de voz, clasificación de imágenes, detección de objetos, y análisis de datos en tiempo real. Esta flexibilidad permite a los desarrolladores explorar y implementar soluciones de IA en múltiples dominios, ampliando las posibilidades de innovación en aplicaciones embebidas e IoT.

En resumen, esp-tflite-micro proporciona una plataforma potente y eficiente para ejecutar modelos de inteligencia artificial en microcontroladores ESP32, optimizando el uso de recursos limitados, facilitando la integración con el entorno de desarrollo ESP-IDF, y ofreciendo una gran flexibilidad para adaptarse a diversas aplicaciones y modelos de IA.

Descripción del proyecto

La detección de personas en imágenes es una aplicación común de la inteligencia artificial (IA), donde el objetivo es determinar si una imagen contiene a una persona. En este proyecto, utilizaremos la cámara incorporada al dispositivo ESP32-S3 para implementar un sistema de detección de personas en tiempo real.

La ESP32-S3 es una potente plataforma de microcontroladores equipada con capacidades de procesamiento de IA y una cámara integrada, lo que la convierte en una excelente opción para proyectos de visión por computadora embebida. Con el uso de modelos de aprendizaje profundo optimizados para dispositivos de bajo consumo, es posible realizar tareas complejas como la detección de personas de manera eficiente.

Estructura general

Este programa de detección de personas está basado en el ejemplo proporcionado por la librería esp-tflite-micro. Está diseñado con un sistema de tareas que se ejecutan de manera concurrente para la recolección de imágenes como campos de datos, su procesamiento a través del modelo de inteligencia artificial, y la generación de respuestas en base a los resultados obtenidos. Todo esto se inicia en un archivo principal main.cc que ejecuta una tarea donde, primero, encontramos una función setup, y después, un bucle while con la función loop.

Todo esto se ejecuta en un sistema operativo de tiempo real (RTOS), lo que garantiza la gestión eficiente y oportuna de las tareas concurrentes, asegurando que el sistema responda adecuadamente a las demandas de procesamiento en tiempo real. Esto permite que el ejemplo de detección de personas funcione de manera fluida y eficiente, proporcionando resultados precisos y rápidos.

Función setup()

La función **setup()** es una función de inicialización crítica en el programa de detección de personas. Su propósito es preparar todos los componentes necesarios para que el sistema de detección de personas funcione correctamente. A continuación, se describe detalladamente cada paso de esta función:

1. Mapeo del modelo de IA:

```cpp
model = tflite::GetModel(g_person_detect_model_data);
if (model->version() != TFLITE_SCHEMA_VERSION) {
  MicroPrintf("Model provided is schema version %d not equal to supported version %
  return;
}
```

El modelo de detección de personas se mapea en una estructura de datos utilizable. Este proceso no implica copia o análisis, por lo que es una operación muy ligera. Se verifica si la versión del modelo coincide con la versión del esquema soportado. Si no coinciden, se imprime un mensaje de error y la función termina.

2. Asignación de memoria para el tensor_arena:

```cpp
if (tensor_arena == NULL) {
  tensor_arena = (uint8_t *) heap_caps_malloc(kTensorArenaSize, MALLOC_CAP_INTERNAL
}
if (tensor_arena == NULL) {
  printf("Couldn't allocate memory of %d bytes\n", kTensorArenaSize);
  return;
}
```

Se asigna memoria para tensor_arena, que es el área de memoria utilizada para los tensores del modelo. Si la asignación de memoria falla, se imprime un mensaje de error y la función termina.

3. Creación del intérprete:

```cpp
static tflite::MicroInterpreter static_interpreter(model, micro_op_resolver, tensor
interpreter = &static_interpreter;
```

Se crea un intérprete estático para manejar la inferencia del modelo, utilizando el modelo cargado, el resolutor de operaciones y el área de memoria asignada.

4. Asignación de tensores:

```cpp
TfLiteStatus allocate_status = interpreter->AllocateTensors();
if (allocate_status != kTfLiteOk) {
  MicroPrintf("AllocateTensors() failed");
  return;
}
```

Se asigna memoria para los tensores del modelo desde tensor_arena. Si la asignación falla, se imprime un mensaje de error y la función termina.

5. Obtención de información sobre el área de entrada del modelo:

```cpp
input = interpreter->input(0);
```

Se obtiene información sobre el área de memoria que se utilizará para la entrada del modelo.

6. Inicialización de la cámara:

```cpp
#ifndef CLI_ONLY_INFERENCE
TfLiteStatus init_status = InitCamera();
if (init_status != kTfLiteOk) {
  MicroPrintf("InitCamera failed\n");
  return;
}
#endif
```

Se inicializa la cámara para capturar imágenes. Si la inicialización falla, se imprime un mensaje de error y la función termina.

7. Inicialización del sistema RTOS:

```cpp
ESP_ERROR_CHECK(nvs_flash_init());
ESP_ERROR_CHECK(esp_netif_init());
ESP_ERROR_CHECK(esp_event_loop_create_default());
```

Se inicializa la cámara para capturar imágenes. Si la inicialización falla, se imprime un mensaje de error y la función termina.

8. Inicialización de WiFi y MQTT:

```cpp
wifi_init();
vTaskDelay(5000 / portTICK_PERIOD_MS);
mqtt_service_init(MQTT_BROKER_URL);
vTaskDelay(5000/portTICK_PERIOD_MS);
```

Se inicializa la cámara para capturar imágenes. Si la inicialización falla, se imprime un mensaje de error y la función termina.

Función loop()

La función loop() es la encargada de capturar imágenes, ejecutar el modelo de detección de personas, procesar los resultados y responder a la detección. Se ejecuta en un bucle infinito, realizando estas tareas de forma continua. A continuación se describe detalladamente cada paso de esta función:

1. Captura de la imagen:

```cpp
if (kTfLiteOk != GetImage(kNumCols, kNumRows, kNumChannels, input->data.int8)) {
  MicroPrintf("Image capture failed.");
}
```

Se captura una imagen utilizando la función GetImage, que llena el tensor de entrada del modelo con los datos de la imagen capturada. Si la captura de la imagen falla, se imprime un mensaje de error.

2. Ejecución del modelo:

```cpp
if (kTfLiteOk != interpreter->Invoke()) {
  MicroPrintf("Invoke failed.");
}
```

Se ejecuta el modelo de inteligencia artificial en la imagen capturada llamando a interpreter->Invoke(). Si la ejecución del modelo falla, se imprime un mensaje de error.

3. Obtención de los resultados de inferencia:

```cpp
TfLiteTensor* output = interpreter->output(0);
```

Se obtiene el tensor de salida del modelo, que contiene los resultados de la inferencia.

4. Procesamiento de los resultados de inferencia:

```cpp
int8_t person_score = output->data.uint8[kPersonIndex];
int8_t no_person_score = output->data.uint8[kNotAPersonIndex];

float person_score_f = (person_score - output->params.zero_point) * output->params.
float no_person_score_f = (no_person_score - output->params.zero_point) * output->p
```

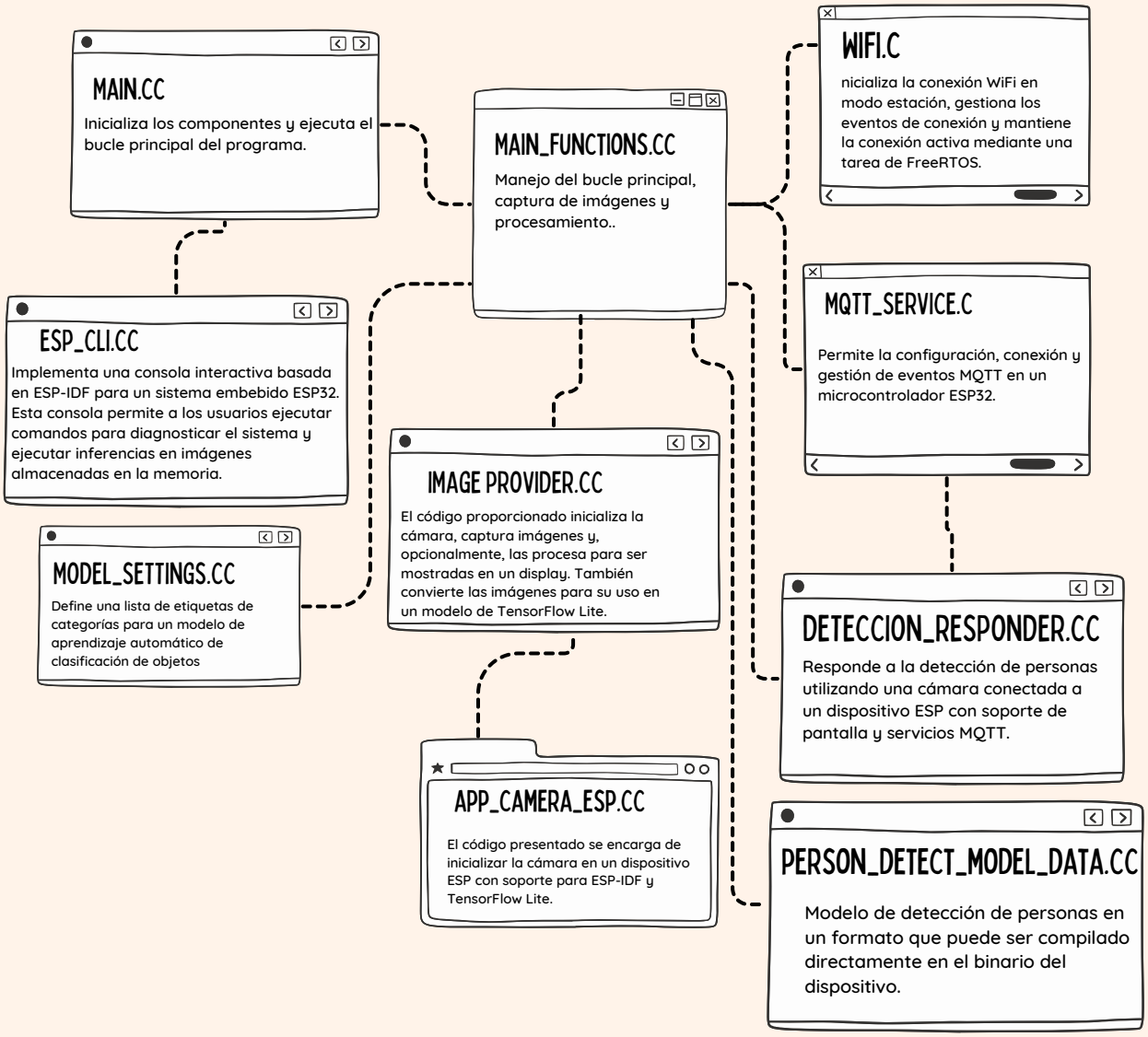
Se extraen los puntajes correspondientes a las categorías "persona" y "no persona" del tensor de salida. Estos puntajes se ajustan utilizando el punto cero y la escala del tensor para convertirlos a valores de punto flotante.

4. Respuesta a la detección:

```cpp
RespondToDetection(person_score_f, no_person_score_f);
```

Se introduce un retraso de 1 segundo para evitar que el watchdog del sistema se dispare. Esto asegura que el bucle loop no se ejecute demasiado rápido, permitiendo que el sistema se mantenga estable.

Diagrama de ficheros



Anexos

Anexo I: Programación de redes neuronales

Programación de las IA
La programación de las IA se basa en la creación de neuronas, para esto se usarán las clases.

Clase neurona
Creación de una clase en la que hay una serie de parámetros que serán específicos para cada caso. Las neuronas están conectadas entre sí, pero hay diferentes tipos de conexiones unos más complejos y otras más simples.

Tipos de conexiones entre neuronas
Antes de definir los tipos de conexiones hay que entender las diferentes "capas" que tienen las neuronas, para la explicación se utilizarán personas en lugar de neuronas para lograr un mayor entendimiento. Tipos de capas:

- **Capa de entrada**: Es la primera persona o grupo de personas que reciben la información inicial.
- **Capas ocultas**: Son las personas en el medio del grupo que reciben información, la procesan, y la pasan a otros. Pueden ser muchas capas de personas haciendo cálculos.
- **Capa de salida**: Es la última persona o grupo de personas que dan la respuesta final.

Ya explicadas las capas que forman las conexiones neuronales, vamos a profundizar en las diferentes conexiones entre neuronas.

Las conexiones entre neuronas en redes neuronales artificiales son los enlaces a través de los cuales se transmite y procesa la información. Estas conexiones permiten que las redes neuronales aprendan a reconocer patrones y realicen tareas complejas, como clasificación de imágenes, reconocimiento de voz y predicciones basadas en datos. Hay diferentes tipos:

1. Conexión densa o totalmente conectada (Fully Connected Layer):
- **Capa de entrada**: El primer grupo de personas recibe la información inicial.
- **Capas ocultas**: Cada persona en una capa oculta se comunica con todas las personas de la siguiente capa oculta. Esto significa que cada persona en cada capa oculta tiene una conversación con todas las personas de la siguiente capa.
- **Capa de salida**: Finalmente, todas las personas en la última capa oculta hablan con todas las personas en el grupo de salida para dar la respuesta final.

2. Conexión convolucional (Convolutional Layer):
- **Capa de entrada**: El primer grupo de personas recibe la información inicial.
- **Capas ocultas**: En lugar de hablar con todos, cada persona en una capa oculta solo se comunica con un pequeño grupo de vecinos cercanos (un filtro) y esta pequeña conversación se va moviendo por el grupo de personas. Así, cada grupo pequeño de personas en una capa habla con un grupo pequeño en la siguiente capa.
- **Capa de salida**: La última capa oculta pasa la información al grupo de salida de manera similar, asegurándose de que cada pequeña parte de la información se procese localmente antes de la salida final.

3. Conexión recurrente (Recurrent Layer):

- **Capa de entrada:** El primer grupo de personas recibe la información inicial.
- **Capas ocultas: Ca**da persona en una capa oculta no solo habla con la siguiente persona en la siguiente capa, sino que también se recuerda a sí misma lo que dijo en el paso anterior. Esto significa que hay una conversación continua en el tiempo dentro de cada persona.
- **Capa de salida:** Finalmente, las personas en la última capa oculta, que han estado recordando información de pasos anteriores, hablan con el grupo de salida.

4. Conexión de capa residual (Residual Layer):

- **Capa de entrada:** El primer grupo de personas recibe la información inicial.
- **Capas ocultas:** En lugar de hablar solo con la siguiente persona en la siguiente capa, algunas personas tienen una línea directa de comunicación con personas varias capas más adelante, saltándose algunas capas intermedias. Esto asegura que la información importante no se pierda y llegue más rápidamente a capas posteriores.
- **Capa de salida:** Las personas en la última capa oculta, que han recibido información tanto directa como indirectamente, hablan con el grupo de salida.

5. Conexión de atención (Attention Layer):

- **Capa de entrada:** El primer grupo de personas recibe la información inicial.
- **Capas ocultas:** Cada persona en una capa oculta decide a quién escuchar con más atención en la capa anterior, en lugar de escuchar a todos por igual. Esto significa que algunas personas pueden decidir que la información de ciertas personas en la capa anterior es más relevante.
- **Capa de salida:** Finalmente, las personas en la última capa oculta, habiendo prestado atención selectiva a la información importante, hablan con el grupo de salida.

Las conexiones entre neuronas tienen diferentes usos dependiendo de su complejidad.
Las redes más complejas permiten aprender patrones más complejos, rinden mejor en tares pesadas y se adaptan mejor a cualquier tipo de problemas.

A su vez las redes más sencillas tienen un menor riesgo de sobreajuste, requieren menos recursos y son más fáciles de entender y alterar por el usuario.

Anexo II: Aprendizaje automático

El aprendizaje automático es un subcampo de la IA que permite a las máquinas aprender de los datos. Los algoritmos de aprendizaje automático se dividen en tres categorías principales: Aprendizaje supervisado:

1. El aprendizaje supervisado: es una técnica de aprendizaje automático donde un modelo se entrena utilizando un conjunto de datos etiquetados. Este conjunto de datos incluye entradas (features) y salidas (labels) conocidas.

El objetivo del modelo es aprender a mapear las entradas a las salidas correctas, de modo que pueda predecir la salida correcta para nuevas entradas. Como las respuestas (labels) son conocidas, para guiar al algoritmo y mejorar su efectividad ante nuevas entradas, se genera una función de perdida (loss function), una función que mide la diferencia entre las predicciones del modelo y las etiquetas reales. Combinado con un algoritmo de optimización que ajuste los parámetros del modelo para minimizar la función de perdida. El algoritmo de optimización más utilizado es el gradiente descente (gradient descent) que funciona calculando el gradiente (la derivada) de la función de pérdida con respecto a los parámetros.
Ejemplos de algoritmos de aprendizaje supervisado incluyen la regresión lineal, los árboles de decisión y las redes neuronales que se explicaran luego detalladamente.

2. Aprendizaje no supervisado: El aprendizaje no supervisado es un tipo de aprendizaje automático en el que un modelo se entrena utilizando datos que no están etiquetados. A diferencia del aprendizaje supervisado, en el cual cada ejemplo de entrenamiento viene con una etiqueta o respuesta correcta, el aprendizaje no supervisado debe descubrir patrones y estructuras en los datos por sí mismo. Por lo que no hay una respuesta correcta para las entradas. Entre los algoritmos de aprendizaje no supervisados, destacan los siguientes:

- Clustering: el objetivo es agrupar los datos en grupos o clústeres basados en la similitud de elementos.

- Reducción de Dimensionalidad: el objetivo es reducir el número de características en los datos mientras se conserva la mayor cantidad posible de información relevante.

- Modelos Generativos: el objetivo es modelar la distribución de datos y generar nuevos ejemplos que sean similares a los datos originales, comúnmente usado en fotografías.

3. Aprendizaje por refuerzo: Este enfoque se basa en que un agente aprende a tomar decisiones mediante la interacción con un entorno. El objetivo es maximizar una recompensa acumulada a lo largo del tiempo. A diferencia del aprendizaje supervisado, donde el modelo aprende de ejemplos etiquetados, el aprendizaje por refuerzo se basa en la retroalimentación obtenida de las acciones tomadas por el agente.

Anexo III: Operaciones Matemáticas

Instrucciones Vectoriales

Las instrucciones vectoriales, también conocidas como instrucciones SIMD (Single Instruction, Multiple Data), son un tipo de operación en la arquitectura de un procesador que permite la ejecución de una misma instrucción sobre múltiples datos simultáneamente. Esto se logra mediante el uso de registros vectoriales que pueden contener varios elementos de datos, en lugar de un único valor.

Características de las instrucciones vectoriales:

- **Paralelismo a nivel de datos:** Permiten aplicar la misma operación aritmética o lógica a varios datos en una sola instrucción, lo que acelera el procesamiento.
- **Registros vectoriales:** Utilizan registros especiales que pueden almacenar varios datos (por ejemplo, múltiples números enteros o de punto flotante) en una sola estructura.
- **Aplicaciones:** Son comunes en aplicaciones que requieren la manipulación de grandes cantidades de datos, como el procesamiento de imágenes, gráficos, audio y aplicaciones científicas.

Ejemplo: Si se tiene una instrucción vectorial que suma dos vectores de números, en lugar de realizar una suma por cada par de elementos de los vectores, la instrucción vectorial puede sumar varios pares simultáneamente, reduciendo el número total de instrucciones ejecutadas.

Procesamiento en Paralelo

El procesamiento en paralelo es una técnica de computación en la que múltiples procesadores o núcleos de procesamiento ejecutan varias tareas o partes de una tarea simultáneamente, con el fin de aumentar la velocidad y eficiencia del procesamiento. Características del procesamiento en paralelo:

- **Distribución de tareas:** Una tarea grande se divide en sub-tareas más pequeñas, que pueden ejecutarse simultáneamente en diferentes núcleos o procesadores.
- **Eficiencia:** Mejora la eficiencia y reduce el tiempo de ejecución total de las tareas, especialmente en aplicaciones que requieren cálculos intensivos.
- **Escalabilidad:** El rendimiento puede mejorar al agregar más procesadores o núcleos, aunque esto depende de cómo se dividan las tareas y de la arquitectura del sistema.

Tipos de paralelismo:

- **Paralelismo a nivel de instrucción (ILP):** Ejecutar múltiples instrucciones de una única secuencia de instrucciones de manera simultánea.
- **Paralelismo a nivel de datos (DLP):** Aplicar la misma operación a múltiples datos al mismo tiempo (similar a las instrucciones vectoriales).
- **Paralelismo a nivel de tareas (TLP):** Ejecutar diferentes tareas o procesos en paralelo.

Ejemplo: En una simulación de clima, el modelo del clima puede dividirse en diferentes regiones geográficas, y cada región puede ser procesada simultáneamente en diferentes núcleos del procesador, acelerando significativamente la simulación completa.

Anexo VI: Librerías Espressif

esp-who
La librería **esp-who** de Espressif es una solución integral para aplicaciones de reconocimiento facial y visión por computadora en los microcontroladores ESP32. Proporciona herramientas y módulos para capturar, procesar y analizar imágenes de manera eficiente. Soporta diversas cámaras compatibles con ESP32, facilitando la configuración y captura de imágenes. Incluye algoritmos para el preprocesamiento de imágenes y ofrece modelos preentrenados para detección y reconocimiento facial, además de módulos para el entrenamiento de estos modelos. La integración con TensorFlow Lite permite ejecutar inferencias de redes neuronales directamente en el ESP32.

Enlace: https://github.com/espressif/esp-who

esp-dl
La librería **esp-dl** es una colección de primitivas y herramientas optimizadas para el desarrollo y ejecución de modelos de deep learning en los microcontroladores ESP32. Mejora la eficiencia del hardware del ESP32 en tareas de procesamiento intensivo. Proporciona funciones básicas para construir redes neuronales profundas, optimizadas para el ESP32. Incluye modelos preentrenados para tareas comunes de visión por computadora, permitiendo una rápida implementación de soluciones. La integración con TensorFlow Lite facilita la ejecución de modelos de aprendizaje automático en dispositivos con recursos limitados.

Enlace: https://github.com/espressif/esp-dl

esp-nn
La librería **esp-nn** (Espressif Neural Network) ofrece optimizaciones y primitivas diseñadas para ejecutar operaciones de redes neuronales en los microcontroladores ESP32 de manera eficiente. Contiene versiones optimizadas de operaciones comunes en redes neuronales, como convoluciones y multiplicaciones de matrices, maximizando el rendimiento y minimizando el uso de memoria. Está diseñada para trabajar con TensorFlow Lite Micro, permitiendo la ejecución eficiente de modelos en dispositivos con recursos limitados. Aprovecha las capacidades de procesamiento en paralelo del ESP32 para mejorar el rendimiento en tareas de inferencia de modelos de deep learning.

Enlace: https://github.com/espressif/esp-nn

Inteligencia Artificial Empotrada

Las Inteligencias Artificiales (IA) simulan el pensamiento humano y realizan tareas que normalmente requieren intervención humana, superando en muchos casos la velocidad y precisión humana. Su uso ha crecido exponencialmente, integrándose en la vida cotidiana con asistentes virtuales, sistemas de recomendación y aplicaciones en medicina, educación y transporte. En programación, la IA facilita la traducción de código, el desarrollo de funciones automatizadas y el análisis de código existente. Su capacidad para manejar grandes volúmenes de datos y ejecutar tareas complejas la convierte en una herramienta esencial. Además, la IA tiene un futuro prometedor en sistemas empotrados como el ESP32S3, donde su eficiencia y capacidad de procesamiento permiten la implementación de soluciones avanzadas de IoT, visión por computadora y reconocimiento de patrones, abriendo nuevas posibilidades en la tecnología integrada y portátil.

Jose Luis Galán | Alberto Andrés | Andrés Toledo
Gonzalo Albelda | Gonzalo Martín