Proyecto Final
Curso de Sistemas Operativos y Laboratorio

Título del proyecto: Reconocimiento de palabras clave en TinyML

Miembros del equipo:

Juan Pablo Gómez López - 1037665653
Danilo Antonio Tovar Arias - 1193578670

Resumen

Este proyecto consiste en el desarrollo de un sistema de reconocimiento de palabras clave utilizando TinyML con TensorFlow Lite en una placa Arduino. El objetivo es implementar un modelo de aprendizaje automático optimizado para dispositivos con recursos limitados, que pueda identificar comandos de voz simples, como “enciende” o “apaga”. Esta solución permite controlar dispositivos mediante voz, sin necesidad de conexión a internet, lo que la hace ideal para aplicaciones embebidas e IoT.

Introducción

- ¿Cuál es la necesidad y/o problema que aborda el desafío seleccionado?

En el panorama tecnológico actual, existe una creciente demanda de dispositivos inteligentes capaces de interactuar de manera más natural y eficiente con los usuarios y su entorno. Sin embargo, muchos de estos dispositivos operan en ubicaciones remotas, con restricciones de energía, conectividad y recursos computacionales. La necesidad principal que aborda este desafío es la de habilitar capacidades de procesamiento de lenguaje natural y control por voz en

dispositivos embebidos de bajo costo y bajo consumo, como los microcontroladores Arduino, sin depender de una conexión constante a la nube para el procesamiento de datos.

Específicamente, el problema a resolver es la latencia inherente y la dependencia de la conectividad en soluciones basadas en la nube para el reconocimiento de voz. Al procesar las palabras clave directamente en el dispositivo (en el "borde"), se eliminan los retrasos de transmisión de datos, se mejora la privacidad al no enviar datos de voz a servidores externos y se reduce significativamente el consumo de energía al no requerir transmisiones de datos continuas. Esto permite desarrollar sistemas de control por voz o de activación por palabra clave que son autónomos, fiables y eficientes, ideales para aplicaciones donde la conectividad es intermitente o inexistente, o donde la respuesta en tiempo real es crítica.

- ¿Por qué es importante el desarrollo de este desafío en el contexto tecnológico actual?

El desarrollo de este desafío es de suma importancia en el contexto tecnológico actual por varias razones fundamentales que se alinean con las tendencias emergentes en la industria:

- **Proliferación del Internet de las Cosas (IoT) y Computación en el Borde (Edge Computing):** Con miles de millones de dispositivos IoT esperados, la capacidad de procesar datos localmente es crucial. TinyML permite que estos dispositivos sean más inteligentes, reactivos y eficientes al no tener que enviar todos los datos a la nube para su análisis. El reconocimiento de palabras clave en el borde es un habilitador clave para interfaces de usuario más intuitivas en el IoT.
- **Eficiencia Energética y Sostenibilidad:** Los dispositivos Arduino, por su naturaleza de bajo consumo, se benefician enormemente de TinyML. Al ejecutar inferencias de aprendizaje automático directamente en el microcontrolador, se reduce drásticamente el consumo de energía en comparación con las soluciones basadas en la nube, lo que prolonga la vida útil de las baterías y contribuye a sistemas más sostenibles.
- **Privacidad y Seguridad de los Datos:** El procesamiento de voz en el dispositivo garantiza que los datos de audio sensibles no salgan del mismo, mejorando significativamente la privacidad del usuario. En un mundo donde la seguridad de los datos es una preocupación creciente, esta capacidad es invaluable.
- **Democratización de la Inteligencia Artificial:** TinyML hace que la inteligencia artificial sea accesible para una gama mucho más amplia de desarrolladores e ingenieros, incluso aquellos sin experiencia en hardware de alto rendimiento. Esto impulsa la innovación y permite que la IA se integre en productos y soluciones que antes eran inviables debido a las limitaciones de costo y recursos.
- **Nuevas Oportunidades en Aplicaciones de Bajo Costo:** El reconocimiento de palabras clave en Arduino abre la puerta a una infinidad de aplicaciones de bajo costo en domótica, agricultura inteligente, monitoreo industrial, dispositivos médicos portátiles, juguetes interactivos y mucho más, donde la integración de capacidades de voz en un factor de forma pequeño y económico es un diferenciador clave.

Antecedentes o marco teórico

La comprensión y ejecución de este proyecto de reconocimiento de palabras clave con TinyML en Arduino requieren la integración de conocimientos provenientes de diversas áreas. Los principales aspectos teóricos necesarios abarcan desde el aprendizaje automático y el procesamiento de señales hasta la arquitectura de microcontroladores y los principios de los sistemas operativos embebidos.

- ¿Cuáles son los principales aspectos teóricos necesarios para comprender el desafío y llevarlo a cabo?

Para comprender y llevar a cabo este desafío, es fundamental tener un conocimiento sólido de los siguientes aspectos teóricos:

- Aprendizaje Automático (Machine Learning):
    - Conceptos Fundamentales: Entender qué es el aprendizaje supervisado (dado que el reconocimiento de palabras clave se entrena con datos etiquetados), los conceptos de entrenamiento, validación y prueba de modelos.
    - Redes Neuronales Convolucionales (CNNs): Son la arquitectura de red neuronal predominante para el procesamiento de datos de tipo grid, como imágenes y, en este caso, espectrogramas de audio. Se debe comprender cómo las capas convolucionales, de pooling y densas extraen características y clasifican patrones.
    - Clasificación: Saber cómo un modelo de Machine Learning asigna una entrada (en este caso, un segmento de audio) a una de varias categorías predefinidas (las palabras clave o el "ruido").
    - Métricas de Evaluación: Familiaridad con métricas como la precisión (accuracy), la sensitividad (recall) y la especificidad para evaluar el rendimiento del modelo.
- Procesamiento Digital de Señales (DSP) para Audio:
    - Muestreo y Cuantificación: Entender cómo las ondas de sonido analógicas se convierten en señales digitales (PCM).
    - Transformada Rápida de Fourier (FFT): Esencial para convertir la señal de audio del dominio del tiempo al dominio de la frecuencia. Esto permite extraer características espectrales del sonido, que son mucho más relevantes para el reconocimiento de voz que la forma de onda cruda.
    - Espectrogramas y MFCCs (Mel-Frequency Cepstral Coefficients): Los espectrogramas visualizan la energía de las frecuencias a lo largo del tiempo. Los MFCCs son características aún más compactas y robustas que imitan la

percepción auditiva humana, siendo comúnmente utilizadas como entrada para modelos de reconocimiento de voz.
    - Pre-procesamiento de Audio: Técnicas como la normalización, eliminación de ruido y detección de actividad de voz (VAD) para mejorar la calidad de los datos de entrada al modelo.
- TinyML y TensorFlow Lite:

    - Concepto de TinyML: Comprender que se trata de la aplicación de Machine Learning en dispositivos de muy bajo consumo y recursos limitados.
    - TensorFlow Lite: Conocer su propósito como *framework* para desplegar modelos de TensorFlow en el borde. Entender el proceso de cuantificación (reducir la precisión de los pesos y activaciones del modelo) para reducir el tamaño del modelo y acelerar la inferencia en hardware limitado.
    - TensorFlow Lite for Microcontrollers: Específicamente diseñado para microcontroladores, requiere la integración de una librería reducida y la adaptación del modelo.
    - Inferencia en el Borde: El concepto de ejecutar el modelo entrenado directamente en el dispositivo en lugar de en la nube.
- Arquitectura de Microcontroladores (Arduino):

    - Componentes Básicos: CPU (o MCU), memoria (Flash para programa, SRAM para datos), periféricos (GPIOs, ADC/DAC, temporizadores, UART/SPI/I2C).
    - Limitaciones de Recursos: Comprender las restricciones severas en términos de velocidad de reloj, tamaño de la memoria RAM y Flash, y capacidad de cómputo, lo que justifica la necesidad de TinyML.
    - Interacción con Periféricos: Cómo leer datos del micrófono (generalmente a través de un ADC) y cómo interactuar con otros componentes (LEDs, actuadores) para la respuesta del sistema.
- Programación de Sistemas Embebidos (C/C++):

    - Programación a Bajo Nivel: Familiaridad con la programación en C/C++ para interactuar directamente con el hardware y optimizar el código para eficiencia.
    - Manejo de Memoria: Conciencia sobre el uso eficiente de la memoria RAM y Flash, dado que los recursos son limitados.
    - Bucles de Eventos y Manejo de Interrupciones: Conceptos clave para sistemas reactivos y eficientes en microcontroladores.

- ¿Qué relación tiene esta teoría con los temas del curso de Sistemas Operativos (en la parte teórica y para el laboratorio)?

Aunque Arduino, en su configuración básica, no ejecuta un sistema operativo completo, este proyecto se entrelaza profundamente con principios clave de los Sistemas Operativos. La

gestión de memoria es crítica, ya que debemos optimizar el uso de la RAM y la Flash del microcontrolador, evitando desbordamientos, un concepto que se estudia a fondo en la teoría de SO. La necesidad de procesar audio en tiempo real y realizar inferencias del modelo se relaciona con la gestión de procesos/tareas y la concurrencia, donde el uso eficiente de interrupciones para el muestreo de audio imita la forma en que los SO manejan eventos asíncronos y E/S. La gestión de E/S se manifiesta en la interacción directa con el Conversor Analógico-Digital (ADC) del micrófono, mientras que la búsqueda de una respuesta rápida y predecible del sistema nos acerca a los conceptos de sistemas de tiempo real y baja latencia. En esencia, el proyecto es un laboratorio práctico donde los desafíos de recursos de los SO se hacen palpables a pequeña escala, obligándonos a aplicar principios de eficiencia y optimización de recursos que son universales en el diseño de sistemas computacionales.

Objetivos (principal y específicos)

- ¿Qué implementación propone realizar?

Objetivo General

Desarrollar e implementar un sistema de reconocimiento de palabras clave de bajo consumo energético y recursos limitados en una plataforma Arduino, utilizando TinyML con TensorFlow Lite.

Objetivos Específicos

1. Entrenar y optimizar un modelo de red neuronal adecuado para la detección de una o dos palabras clave específicas, utilizando técnicas de cuantificación para adaptarlo a las restricciones de memoria de los microcontroladores.
2. Integrar el modelo optimizado en el entorno de desarrollo de Arduino, estableciendo el flujo necesario para la adquisición de audio en tiempo real y su pre-procesamiento en el microcontrolador.
3. Implementar la inferencia del modelo en el dispositivo Arduino para detectar las palabras clave entrenadas y activar una respuesta básica (ej. encender un LED) al reconocerlas.

Metodología

- ¿Cuáles son las principales herramientas que podrían utilizarse para implementar la solución?

Para llevar a cabo este proyecto, se empleará una combinación de *hardware* y *software* específicos. Como plataforma de desarrollo embebido, se utilizará una placa Arduino, preferentemente una con capacidad de procesamiento suficiente para el audio y los modelos de ML, como la Arduino Nano 33 BLE Sense o la Arduino Nicla Voice, que incorporan un micrófono digital y un microcontrolador Cortex-M4 o similar, adecuados para TinyML. En cuanto al *software*, el ecosistema de TensorFlow será central, particularmente TensorFlow Lite para la conversión y optimización de modelos, y TensorFlow Lite for Microcontrollers para la ejecución en el dispositivo.

Para la fase de entrenamiento, se utilizará Python como lenguaje de programación, junto con librerías como TensorFlow/Keras para construir y entrenar las redes neuronales, y librosa o torchaudio (o funciones equivalentes de TensorFlow) para el pre-procesamiento de señales de audio (FFT, MFCCs). Las herramientas de Edge Impulse podrían ser consideradas como una alternativa o complemento, ya que ofrecen una plataforma integrada para la recolección de datos, pre-procesamiento, entrenamiento y despliegue para TinyML, simplificando el flujo de trabajo. Para la programación del Arduino, se empleará el Arduino IDE o PlatformIO, con un enfoque en la programación en C/C++ debido a su eficiencia y control a bajo nivel sobre el *hardware*. Finalmente, un micrófono digital compatible con Arduino (como los que se integran en las placas mencionadas o módulos I2S) será esencial para la adquisición de datos de audio.

- ¿Qué actividades son necesarias para cumplir los objetivos? (Tenga en cuenta el orden lógico para el desarrollo de las actividades).

En primer lugar, la fase de Preparación del Conjunto de Datos es crucial. Esto implica la recolección de audio de las palabras clave deseadas, así como de un conjunto representativo de "ruido" o "fondo" para entrenar al modelo a diferenciar entre las palabras objetivo y el silencio o sonidos irrelevantes. Se buscará obtener una cantidad suficiente de muestras de audio de buena calidad, variando los hablantes y los entornos para mejorar la robustez del modelo. Una vez recolectados, los datos se pre-procesarán, lo que incluye la normalización del volumen, la segmentación en fragmentos de duración fija y la extracción de características espectrales (como MFCCs) que serán la entrada para la red neuronal.

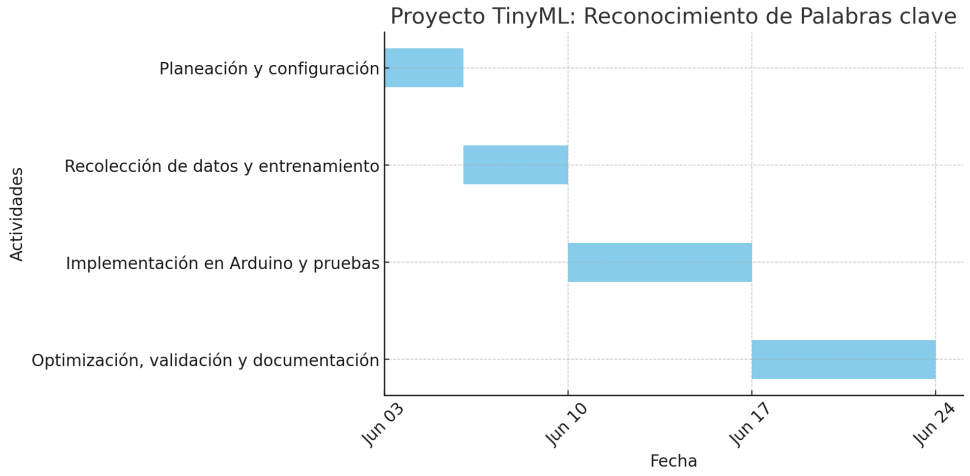
Posteriormente, se abordará el Desarrollo y Entrenamiento del Modelo de Machine Learning. Utilizando las características de audio extraídas, se procederá a diseñar y entrenar una red neuronal convolucional (CNN) adecuada para la clasificación de las palabras clave. Durante este proceso, se experimentará con diferentes arquitecturas y parámetros de red para encontrar un equilibrio entre la precisión del reconocimiento y la complejidad computacional. Un paso fundamental será la cuantificación del modelo (generalmente a 8 bits) utilizando las herramientas de TensorFlow Lite, lo que reducirá drásticamente el tamaño del modelo y acelerará su inferencia, haciéndolo apto para el Arduino. Se realizarán pruebas de validación exhaustivas para asegurar que la cuantificación no degrade significativamente el rendimiento.

La tercera fase se centrará en la Implementación y Optimización en Arduino. Esto implica adaptar el código para el pre-procesamiento de audio al entorno de microcontrolador, replicando las etapas de extracción de características realizadas durante el entrenamiento. Luego, se integrará el modelo cuantificado en el *firmware* del Arduino utilizando la librería TensorFlow Lite for Microcontrollers. Este paso requerirá un manejo cuidadoso de la memoria y la optimización del código C/C++ para asegurar que tanto el modelo como el *framework* encajen en la RAM y Flash disponibles. Se configurará la adquisición de audio en tiempo real desde el micrófono digital, asegurando un muestreo constante y la correcta alimentación de los datos pre-procesados al modelo.

Finalmente, se realizará la fase de Pruebas y Validación en el Dispositivo. Una vez que el sistema esté cargado en el Arduino, se llevarán a cabo pruebas rigurosas en el entorno real para evaluar la precisión del reconocimiento de las palabras clave bajo diversas condiciones de ruido y distancia. Se medirá la latencia del sistema (el tiempo desde que se pronuncia la palabra hasta que se activa la acción) para asegurar una respuesta en tiempo real. La activación de la respuesta básica (ej. encender un LED) servirá como indicador visual de la detección exitosa. Cualquier deficiencia detectada en esta etapa podría conducir a iteraciones en las fases anteriores, ya sea ajustando el pre-procesamiento, refinando el modelo o buscando optimizaciones adicionales en el código del microcontrolador para mejorar el rendimiento.

Cronograma

- Desarrolle un diagrama de Gantt que refleje el desarrollo de las actividades.



Referencias

- TinyML voice recognition: ESP32 vs. Arduino vs. STM32 hardware showdown - DFRobot. (2024, septiembre 22). Dfrobot.com. https://www.dfrobot.com/blog-14005.html
- Rovai), M. (marcelo. (s/f). TinyML made easy: KeyWord spotting (KWS). Hackster.Io. Recuperado el 28 de mayo de 2025, de https://www.hackster.io/mjrobot/tinyml-made-easy-keyword-spotting-kws-5fa6e7
- Vt, A. [@aswinvt]. (s/f). VoiceAI for home automation #TinyML #machinelearning #homeautomation #AI #artificialintelligence. Youtube. Recuperado de: https://www.youtube.com/shorts/Nu3sTY7YM7Q
- Robocraze [@Robocraze]. (s/f). Arduino TinyML Kit Tutorial #7: Keyword spotting using the mic and EdgeImpulse. Youtube. Recuperado de: https://www.youtube.com/watch?v=bEwtH02x_is&ab_channel=Robocraze
- Barovic, A., & Moin, A. (2025). TinyML for speech recognition. arXiv. https://arxiv.org/abs/2504.16213
- Patel, P., Gupta, N., & Gajjar, S. (2023). Real time voice recognition system using Tiny ML on Arduino Nano 33 BLE. En 2023 IEEE International Symposium on Smart Electronic Systems (iSES) (pp. 385–388). IEEE. https://doi.org/10.1109/iSES58672.2023.00085
- Warden, P., & Situnayake, D. (2019). TinyML. O'Reilly Media, Inc.
- Gupta, A. K., Prasad Nandyala, D. S., Nandyala, S. P. (2023). Deep Learning on Microcontrollers: Learn how to develop embedded AI applications using TinyML (English Edition). Germany: Bpb Publications.