🗣️ EdgeAI: Bird Species Detection

📝 Description

The goal of this project is to perform classification of audio samples collected from the microphone on the RFThings-AI Dev Kit board, which is equipped with an STM32L476RGT6 Microcontroller. The target application is bird species detection from songs for the following 3 species: Emberiza cirlus, Muscicapa striata, and Alauda arvensis.

🔄 Workflow

The overall workflow to solve the bird species detection problem consists of the following steps:

1. Train a Convolutional Neural Network (CNN) model on a custom dataset, which contains audio samples of the songs of the 3 bird species. The model is trained in Python using Keras/TensorFlow.
2. Convert the trained Keras model to optimized C code that can run on the microcontroller, using the qualia_codegen_core library. Quantize the model weights and activations to 16-bit fixed-point to reduce memory footprint.
3. Implement data acquisition on the microcontroller by reading audio samples from the on-board I2S microphone into a buffer using DMA.
4. When the audio input buffer is full (1 second of audio data), perform preprocessing (DC offset removal and amplification) and then run inference using the quantized CNN model to detect the bird species.
5. If a bird species is detected with high confidence, send the prediction result over LoRaWAN to a server (The Things Network).

The overall data flow is illustrated in this diagram:

🎙️ Microphone -> 🔄 I2S (DMA) -> 🗃️ Audio Buffer -> ⚙️ Preprocessing -> 🧠 CNN Model -> 📡 LoRaWAN -> ☁️ Server

🧠 CNN Model

The CNN model used is based on the M5 architecture from this paper. It consists of several 1D convolution and pooling layers:

```
Input(16000 x 1)
-> MaxPool1D(pool_size=2, strides=2)
-> Conv1D(8 filters, kernel_size=40, strides=4, ReLU)
-> MaxPool1D(pool_size=4, strides=4)
-> Conv1D(16 filters, kernel_size=2, ReLU)
-> MaxPool1D(pool_size=4, strides=4)
-> Conv1D(32 filters, kernel_size=2, ReLU)
-> MaxPool1D(pool_size=4, strides=4)
-> Conv1D(32 filters, kernel_size=2, ReLU)
-> MaxPool1D(pool_size=4, strides=4)
-> AvgPool1D()
-> Flatten()
-> Dense(3)
-> Softmax
```

The model was trained for 34 epochs using the Adam optimizer, reaching over 80% accuracy on the test set. The trained model was then converted to int16 Q9.7 fixed-point using the qualia_codegen_core library and compiled to C code.

🔌 Microcontroller Processing Architecture

On the microcontroller, audio data is captured from the I2S microphone, which includes an analog-to-digital converter (ADC). The ADC converts the analog audio signal to digital samples.

The I2S interface consists of at least 3 signals:

- Bit clock (BCK) - continuous clock signal that determines speed of data transfer
- Word select (WS) - indicates which channel (left or right) the data is for
- Data (SD) - serial data signal

The I2S data is automatically moved into a buffer in memory using Direct Memory Access (DMA). DMA allows data transfer between peripherals and memory without involving the CPU, enabling the microphone samples to be collected in the background while the CPU performs other tasks.

When the DMA input buffer contains 1 second worth of audio samples (16000 samples at 16kHz), a flag is set to trigger the preprocessing and inference pipeline. The audio data is first processed to remove DC offset and amplify the signal. The processed data is then passed into the quantized CNN model to detect the bird species.

📊 Results

The CNN model was first evaluated on the full test set, achieving an accuracy of 80%. It was then evaluated on a subset of 250 samples, reaching an accuracy of 78%.

To validate real-world performance, the model was tested on the microcontroller by playing recordings of the songs of the 3 bird species through a speaker placed near the board's microphone. The model was able to reliably detect the bird species in 70% of cases. Some misclassifications occurred for similar sounding songs.

Performance metrics:

- Accuracy on full test set: 86%
- Accuracy on 250 sample test subset: 84%
- Accuracy on real bird song recordings: ~75%
- Model memory footprint: 83 KB flash, 20 KB SRAM
- Inference latency: 123 ms
- Estimated power consumption: ~45 mW
- Estimated battery life: 51 hours (assuming 700 mAh battery)

🎯 Conclusion

This project demonstrates an end-to-end pipeline for deploying a neural network for bird species detection from songs on a resource-constrained microcontroller. The CNN model is able to achieve good accuracy on the bird species detection task. Quantization to 16-bit fixed-point allows it to fit within the memory limits of the STM32 MCU.

Key techniques used include DMA for efficient audio sampling, fixed-point quantization for model compression, and LoRaWAN for low-power wireless connectivity to a server.

Some areas for future enhancement:

- Expand the species vocabulary beyond the current 3

- Further optimize the CNN architecture to reduce memory footprint and latency
- Explore duty cycling and other techniques to reduce power consumption
- Add a voice activity detector to only run inference when a song is detected

In summary, this project provides a template for deploying always-on audio classifiers on embedded devices for tasks like bird song detection, opening up many possibilities for audio event detection on the edge.

🔮 What would we had done if we had more time?

- ⬛ Fix the confidence value which is too high sometimes
- ⬛ Implement a circular buffer inside the microphone arduino handler to avoid cuts in the audio stream
- ⬛ Improve the model accuracy while lowering its size
- ⬛ Find why received data on TTN is random

✍️ Authors

Marc Pinet - marcpinet
Arthur Rodriguez - rodriguezarthur
Lucie Andres - lucieandres

📃 License