EEL3090 - Embedded Systems
Project Report

By Om Kumar (B22CS081)

Problem Statement

ES15(Eligible for Bonus): Implement **deepfake detection** technique on a small embedded system(resource-constrained environment).

Team Configuration

This project is done solely by me.

Introduction

I have implemented the deepfake detection task(inference) on the board.
The following tasks were done:

- Fintuned teacher model
- Trained the student model
- Converted the pth file to tflite file format
- Loaded the tflite file into STMCubeIDE
- Generated the image arrays in unit_8 format (an image_data.c file)
- Finally, I wrote the main.c file in STMCubeIDE

Project File: Google drive
Working Video: Video link (with explanation of all tasks done)

Setup information

- Model preparation in the Python environment on my laptop.
- STMCubeIDE for Hardware integartion
- Embedded Board used is **STM32F429ZI** (Discovery Board)

Model Prepration

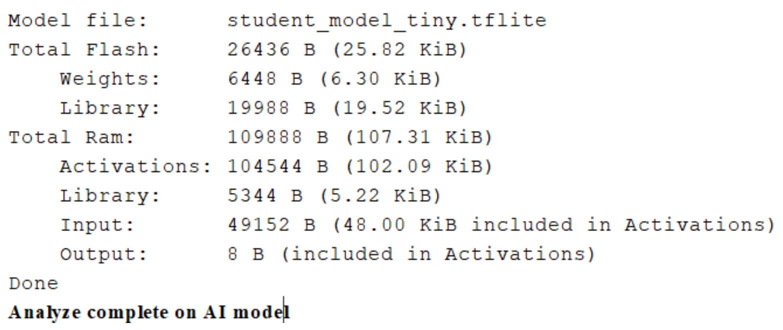
I have used the **Celeb-DF dataset** for training, validation and testing. First of all, a fine-tuned **teacher model**, which is a very big model, i.e ResNet50, for our task of Deepfake detection using the train split of the dataset. Here, the images are resized to 64 x 64 to fulfil the requirement of running the inference on the board.

After this, I used this finetuned teacher model and trained a very light-weight **student model** (with a combination of 3 conv and pool layers and finally a global pooling and very small fully connected layer). This process is known as **Knowledge Distillation**. The acuracy of the teacher model is 99.75% and that of the student model is 87.24% on the validation set and this can be further improved if I ran this for much more epochs.

Then converted the pth file generated into the tflite file (through onnx conversion in between). So, finally got the **student_model_tiny.tflite** file with a size of about 8Kbs only.

Model Loading

I have loaded this model into STMCubeIDE using X-Cube-AI present in this IDE. I haven't used any compression here and kept it balanced between the RAM and time importance options present in the IDE.
Model Statistics:



```
Model file:      student_model_tiny.tflite
Total Flash:     26436 B (25.82 KiB)
    Weights:     6448 B (6.30 KiB)
    Library:     19988 B (19.52 KiB)
Total Ram:       109888 B (107.31 KiB)
    Activations: 104544 B (102.09 KiB)
    Library:     5344 B (5.22 KiB)
    Input:       49152 B (48.00 KiB included in Activations)
    Output:      8 B (included in Activations)
Done
Analyze complete on AI model
```

Figure 1: Model Size statistics

A significant space of the RAM and flash is empty, that are about 145 KB and 2 MB (almost completely empty). Therefore, I am **loading the images into the memory** itself, as each image is about 12Kbs, so I loaded 12 images into the board with the model.

IOC file configuration

Selected the PG13 and PG14 pins as GPIO_output. Clock configuration is automatically set (HCLK = 180MHz) on validating the model(X-Cube-AI).

Code Description

I selected a few images from the test set of the dataset. Then I wrote a Python script to convert these images into the image_data.c file and then loaded this file into the src folder inside the core directory in the Project. Also added the image_data.h file in the inc folder into the core directory containing the 'extern const uint8_t image_1[12288];' statements. Below is a concise, function-by-function walkthrough of my `main.c` implementation, using first-person narrative.

Low-Level Syscalls

At the top of the file, I provide minimal implementations of POSIX-style syscalls (e.g. `_read`, `_write`, `_lseek`, `_fstat`, etc.). These satisfy the STM32's newlib C runtime without requiring a filesystem, returning dummy values or error codes as needed.

System Clock Configuration (`SystemClock_Config`)

Already set as I have configured in the IOC file

Peripheral Initialization

CRC Engine (`MX_CRC_Init`) Already set as I have configured in the IOC file

GPIO (`MX_GPIO_Init`) Already configured in the IOC file. Here used these pins accordingly, means set the PG13 Green for real image and PG14 red for fake image.

Main Application (`main`)

1. **HAL and Clock Setup:** `HAL_Init()` was called, configured the system clock via `SystemClock_Config()`, then initialized GPIO and CRC. (Already written)
2. **AI Network Creation:**
    - Called `ai_network_create()` to build the network handle from the compiled tflite weights.
    - Called `ai_network_init()` to allocate activation buffers and bind the weight data.

3. **Buffer Handles:** Retrieved input and output buffer descriptors using `ai_network_inputs_get()` and `ai_network_outputs_get()`.
4. **Inference Loop:**
   - Iterated over the `NUM_IMAGES` preprocessed frames stored in flash (declared in `image_data.h`).
   - Performed per-pixel normalisation using the ImageNet mean and standard deviation in HWC format.
   - Invoked `ai_network_run()` and verified that exactly one batch is processed.
   - I compared the two outputs ("fake" vs. "real") and lit up the green LED if the real score is higher, otherwise the red LED.
   - I toggle the LEDS with a 500 ms on/off pattern to display each result visibly.
5. **Clean Exit:** After one full pass, I break out of the loop and clear both LEDS.

Utility and Error Functions

`blink_(num)` I wrote this helper to flash the red LED `num` times at 500ms intervals. This is **well-thought-out work to actually debug the code**. As initially there were errors in the code, so I called this function after each line to actually figure out the error based on whether the LED glows or not, and how many times.

`Error_Handler` If any HAL or AI call fails, interrupts are disabled and entered into an infinite loop. Already written function.

Conclusion

I have successfully ported a ResNet50→tiny CNN deepfake detector onto the STM32F429ZI, performing on-chip inference within 256 KB of RAM. The system normalizes each image, runs the model via X-Cube-AI, and provides clear green/red LED feedback, demonstrating deepfake detection under tight resource constraints.

Assumptions

- I relied on the STM32F429ZI's hardware FPU for floating-point operations; no quantisation was applied as the model was loaded into the board without any memory issue.
- I verified that model weights and activation buffers fit within the board's 1MB flash and 256KB RAM while loading the model into the STM32CubeIDE.

Difficulties Faced

- Balancing activation buffer sizes and code/data usage within 256KB RAM. For this, I tried distillation on many student models (about 15 times) and finally figured out the solution by reducing the image size to 64x64, which has significantly decreased the accuracy of the model, but still the accuracy is almost 87%
- Debugging the issues in the main.c file as I have written the code, still no LED glowed, then figured out the issues by using the blink_() function.
- There were issues in the syntax in the other code files in the codebase, which were already written. Correct those issues by moving the AI libraries' inclusion to the top.

References

- Dataset source
- Class notes, lectures and Digital Systems Lab understanding.
- Used perplexity for generating the sample regarding how to write the main.c file. But finally, I have written the main.c file based on the sample code generated. And also used for many instances of debugging.