Hardware Architecture for Deep Learning Homework 3: TinyML test on board

Darpan Gaur CO21BTECH11004

Lakshay Arora CS24RESCH11006

Training tutorial

Setup the STM32CubeIDE, ST-Link, and the board and camera as per the tutorial. Created environment and installed the required packages. Followed the tutorial.

URAT input

- The instructions to send URAT input to the board is not mentioned in the tutorial.
- So, searched for it and found that it can be done using command shell console in STM32CUBEIDE.
- Figure 1 shows the steps to open command shell console.

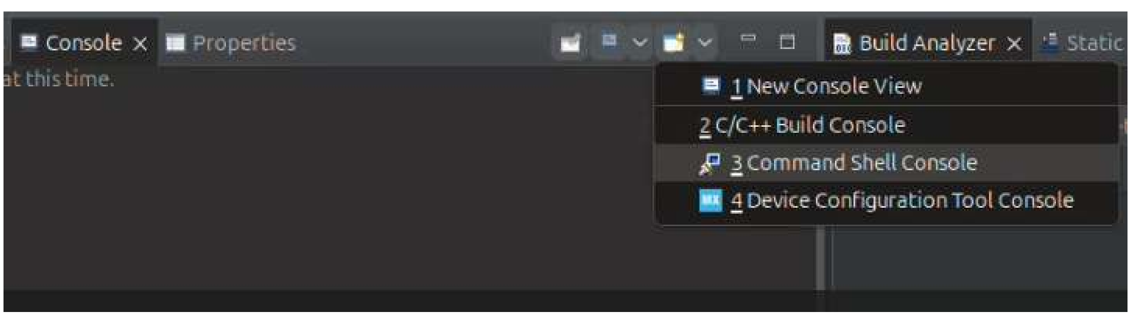

Figure 1: URAT input

- Select connection name as Serial Port. (Figure 2).
- Click on New.. and enter connection name, select the serial port with which board is connected. (Figure 3).
- Finish and click OK.

- Enter "3" then "1" for training class 1.
- Enter "3" then "2" for training class 0.
- Enter "4" for inference.

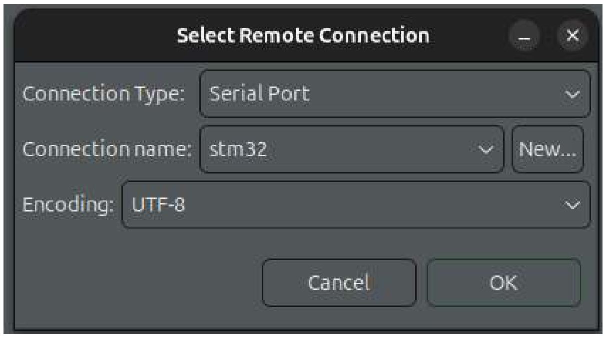

Figure 2: URAT input

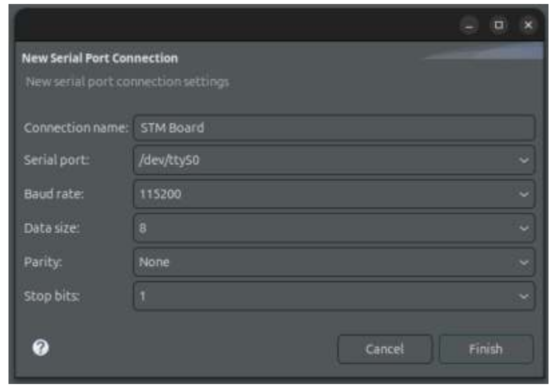

Figure 3: URAT input

Results Training

Figure 4 and 8 shows the training result of class 1 (person).
Figure 5, 9, and 7 shows the training result of class 0 (no person).

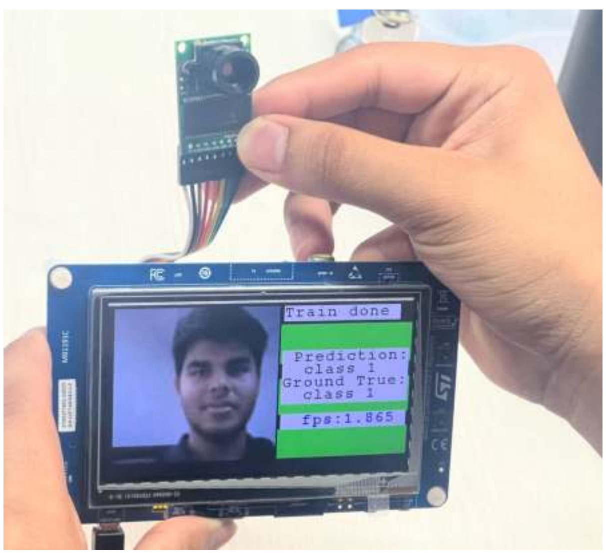

Figure 4: Training class 1

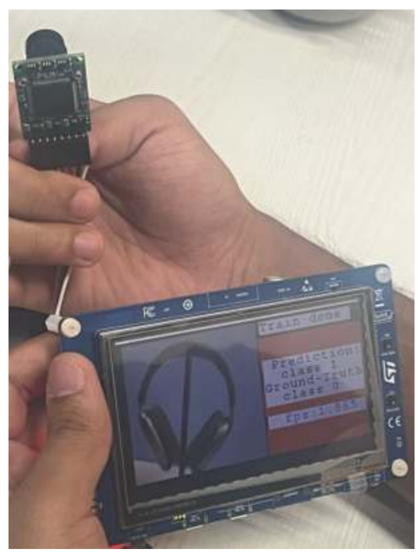

Figure 5: Training class 0

Figure 6: Training class 0

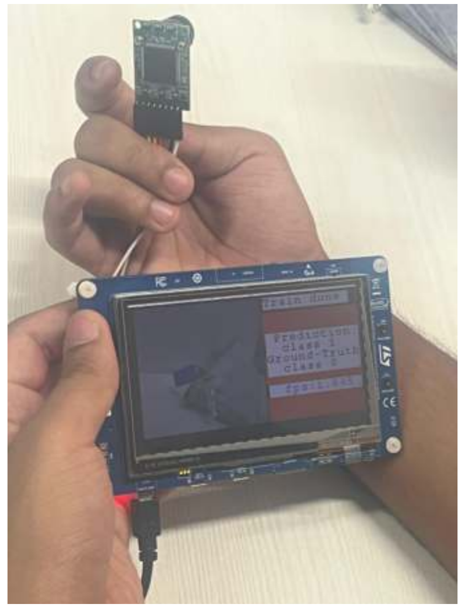

Figure 7: Training class 0

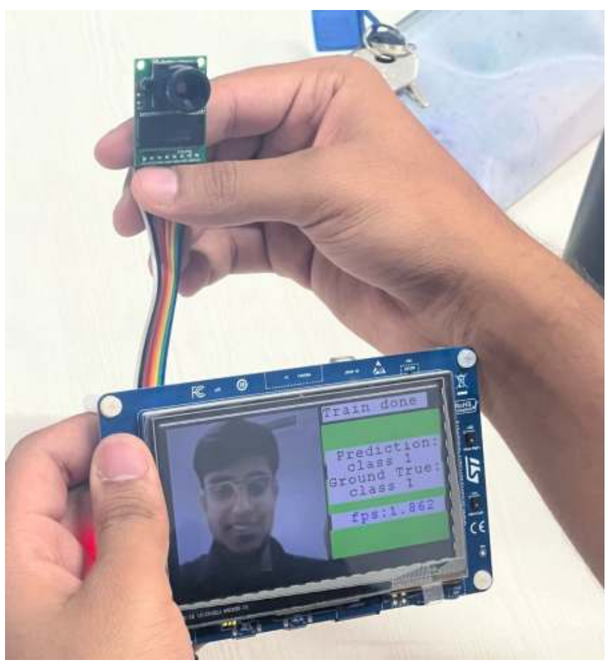

Figure 8: Training class 1

Inference tutorial

Results Inference

Figure 9 shows the inference result of a person without a camera.
Figure 10, 11 shows the inference result of a person with a camera.
Figure 12 shows the inference result of no person.

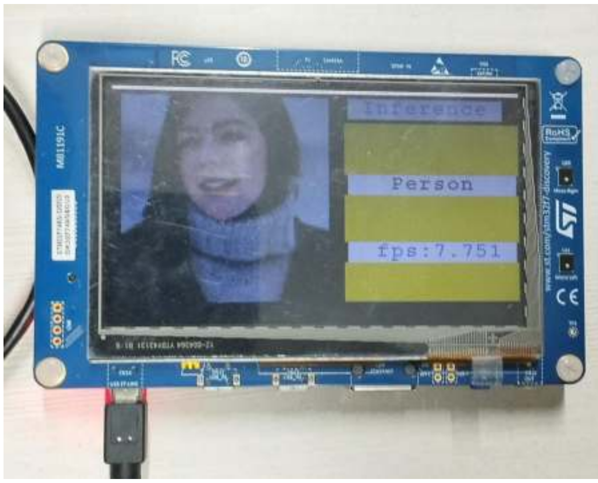

Figure 9: Person (Without camera)

Figure 10: Person

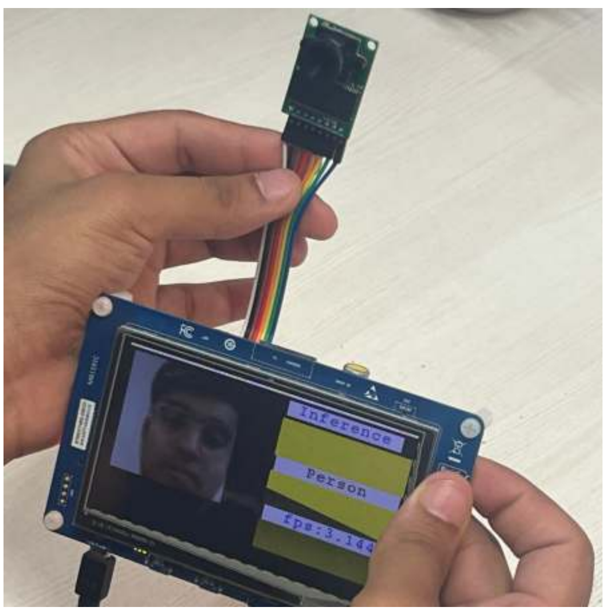

Figure 11: Person

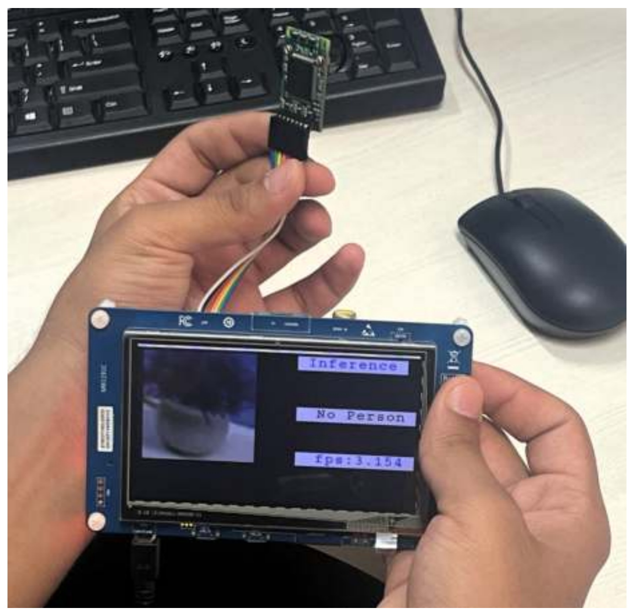

Figure 12: No person

Errors Encountered

- STM32CUBEIDE installations gives error on Ubuntu 23.04 (missing library libncurses5.so). So, installed 24.04, on it works fine (But recommended is LTS 20.04 or 22.04).
- Need to install java and additional libraries for STM32CUBEIDE to work.
- Need to install ST-Link for communication between STM32 board and the host system, not mentioned in the tutorial.
- Tutorial mentioned to use python 3.6, but not able to install it so, used latest version 3.12.3.
- Getting overflow error while preparing the codebase by running python file for both training (tiny_training.py) and testing (www.py), solved by changing data type of the variables in the code. Still some warnings were there while building the codebase.
- Not able to send URAT input to board (see training section).

Learnings

- Embedded Systems Development and Toolchain Setup
  - Gained hands-on experience working with the STM32CubeIDE, flashing firmware onto STM32 boards, and interfacing with external hardware like the Arducam camera.
  - Understood the importance of setting up the development environment properly — including installing required dependencies like Java, ST-Link utilities, and matching compatible OS versions (Ubuntu 24.04 worked better than 23.04, though the recommended was LTS 20.04/22.04).
- Challenges with Legacy Tools and Compatibility
  - Faced difficulties due to incompatibility of Python 3.6 with ARM-based processors and resolved it by using Python 3.12.3, after modifying the code to avoid overflow errors.
  - Learned that open-source repositories, especially older ones, often have missing dependencies, deprecated packages, or incomplete instructions, which require independent troubleshooting.
- Understanding UART and Serial Communication
  - Learned the functioning of UART (Universal Asynchronous Receiver/Transmitter) as a serial communication protocol and how to use the STM32CubeIDE's terminal to send commands to the board.
  - Realized the critical importance of matching the baud rate between transmitter and receiver to ensure reliable data transmission and avoid corruption.

- Hardware Constraints and Memory Management
  - Discovered that flashing our own code erased the factory-installed demo applications. This highlighted the limited storage capacity of the board and the importance of memory management in embedded systems.
  - Learned that real-time inference directly on the device results in significantly lower frames per second (FPS) compared to models trained separately and then deployed — due to computational limitations of microcontrollers.
  - While Training it resulted in 1.865 FPS, whereas while Inference it was 3.144 FPS.
- Hardware-Software Integration and Caution with Components
  - Successfully wired and interfaced the STM32 board with a camera, managing power and signal connections.
  - Realized how easy it is to misconnect wires, potentially damaging delicate components. This taught us the value of patience, precision, and consulting documentation or board schematics.
- Working with Edge AI and On-Device Learning
  - Understood the flow of on-device training vs. inference and the resource constraints (like RAM and CPU) that limit model complexity and speed on microcontrollers.
  - Got practical exposure to deploying AI models on hardware, and the trade-offs involved when choosing model size, accuracy, and inference time.