Proposal for STM32-Based System

Ho Jun Kin
Faculty of Engineering
Universiti Teknologi Malaysia
Skudai 81310 Malaysia
hojunkin@graduate.utm.my

Kevin Ng Zhi Hao
Faculty of Engineering
Universiti Teknologi Malaysia
Skudai 81310 Malaysia
kevinngzhi@graduate.utm.my

Yip Wen Xin
Faculty of Engineering
Universiti Teknologi Malaysia
Skudai 81310 Malaysia
yipwenxin@graduate.utm.my

Abstract - This initiative explores the implementation of image recognition using TensorFlow Lite for Microcontrollers (TFLM) and CMSIS-NN on the Discovery STM32Nucleo-F446RE board. The project is designed to facilitate accuracy and performance testing on any embed-enabled device accommodating the final binary. The model, trained on the MNIST dataset, a database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The methodology involves implementing CMSIS-NN on the STM32F44RE Nucleo board, leveraging its CORTEX-M4 core with SIMD and DSP capabilities. The project processes computer-sourced images on the microcontroller, utilizing the ARM Cortex M4's ART accelerator and FPU over Flash memory. Key steps include neural network configuration, code generation, library integration, and inference execution. The output is transmitted to the computer for tabulation. CMSIS-NN optimization aims to enhance efficiency in neural network inference and image processing tasks, optimizing pixel format conversion for memory efficiency. The Cortex-M4's limitations in memory and cache size contrast with its energy efficiency.

Keywords: TensorFlow Lite, CMSIS-NN, Image Recognition, STM32F44RE, Cortex-M4, MNIST, Embedded Systems, Neural Network, Inference.

I. INTRODUCTION

In the world of embedded systems, the merging of machine learning in microcontroller platform is an exciting and potentially fruitful area for progress and discovery. This initiative aims to demonstrate image recognition using TensorFlow Lite for Microcontrollers (TFLM) and CMSIS-NN specifically tailored for the Discovery STM32nucleo-f446re board. It extends to other embed-enabled devices for comprehensive accuracy and performance testing.

This project builds on a digit image recognition example, adapted to the STM32nucleo-f446re board. The model is trained on the MNIST dataset, which consists of 60,000 training examples and 10,000 test examples of handwritten digits. Our goal is to demonstrate the functionality.

As we navigate through the project, CMSIS_NN which is the Cortex Microcontroller Software Interface Standard - Neural Network, plays a crucial role in enhancing the efficiency and performance of neural network inference on microcontroller platform. It able to provide a set of optimized kernels for common neural network operations, which helps in speeding up the execution of neural network operations on the microcontroller. Besides, CMSIS-NN includes memory management functions that help minimize memory footprint, making it well-suited for devices with limited RAM.

In summary, this project aims to demonstrate the powerful collaboration between machine learning and the Discovery STM32nucleo-f446re board. Leveraging TensorFlow Lite for Microcontrollers and the optimization capabilities of CMSIS-NN, we would like to demonstrate the functionality of this system in this limited-resource environment.

II. PROBLEM / MOTIVATION

Due to resource constraints, applying machine learning for applications like image recognition on small computer devices like the

Discovery STM32nucleo-f446re presents difficulties. The necessity to get over these obstacles and enable effective on-device picture recognition is what drives this effort. Our goal is to close the gap between the capabilities of machine learning and the constraints of microcontroller environments by merging TensorFlow Lite for Microcontrollers and CMSIS-NN. This will open up possible applications in smart sensors and Internet of Things (IoT) devices.

III. LITERATURE REVIEW

By referring to paper [1], the authors in investigates how to build and benchmark a deep neural network (DNN) architecture on a machine that runs on bare metal. The increasing interest in deep neural networks among scientific research groups is the driving force behind this effort because of their adaptability in a wide range of domains. With the advent of integrated hardware and software advancements, DNN models may now be directly implemented on general-purpose microcontrollers, leading to significant system-level cost savings.

Utilizing the capabilities of the STM32F779I-EVAL board, which has an ARM Cortex-M7 core and CMSIS-NN software kernels, is the main goal of the study. The choice of a low-cost and energy-efficient platform indicates the authors' commitment to resolving the practical restrictions associated with embedded systems.

The on-device inference phase, in which a pre-trained DNN produces responses to inputs, is a crucial topic covered in the study. Fast inference on integrated devices, according to the authors, has potential, especially for applications that need to make decisions in real time. Convolutional Neural Networks (CNNs) are emphasized as an important paradigm in the field of DNN architectures, especially suited for computer vision applications where they achieve cutting-edge outcomes. With that, authors decided to use STMicroelectronics board with a low-cost ARM Cortex-M7 core, combined with CMSIS-NN software kernels to proof that microcontroller with limited resources is an alternative that able to implement machine learning.

IV. METHODOLOGY

In this project, the dataset that we used is MNIST dataset. MNIST, which is Modified National Institute of Standards and Technology database, consisting a collection of 70,000 28x28 pixels images of handwritten digits from 0 to 9. There are 60,000 training examples and 10,000 testing examples.

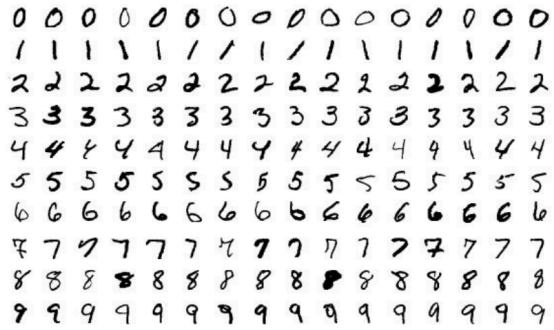

Figure 1: MNIST datasets

This MNIST model implementation chosen for this project is available on GitHub, which builds a convolutional neural network that can achieve over 70% accuracy on the test set examples from the MNIST dataset. This model is written in python and uses the built-in Keras library.

We are using STM32CubeIDE with the STM32 X-Cube-AI tools to train and validate the model into the board. STM32 X-CUBE-AI is a set of libraries and plugins in the IDE. It supports model trained with TensorFlow, Keras, etc. With the tools, we will be able to load the trained model into the microcontroller and proceed with coding to perform inference, which is our image recognition for digit numbers. By applying the

MNIST model into CubeAI, after configuring the model, we will be able to generate a code with the model framework and the MNIST database has been generated to the project.

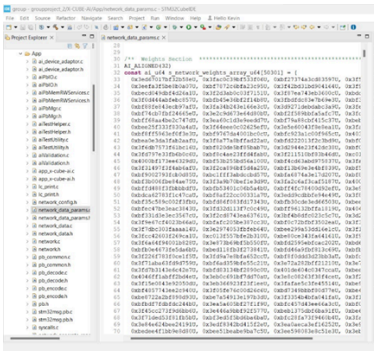

Figure 2: MNIST model in STMCubeIDE

After having the model, we implement the CMSIS-NN (Neural Network) algorithm on the STM32F44RE NUCLEO Development board, which features a CORTEX-M4 core. The CMSIS-NN software library is designed specifically for processors with SIMD capability (CORTEX-M0), DSP extension (CORTEX-M4), and MVE extension (CORTEX-M55). The library provides essential functions such as Convolution, Activation, Fully-connected Layer, SVDF layer, Pooling, Softmax, and Basic Math.

The project involves processing images received from a computer on the STM32F44RE microcontroller using the CMSIS-NN library. To ensure compatibility with the microprocessor, the images are converted to the RGB565 color format, a 16-bit format suitable for efficient processing by the M4 core.

Utilizing CMSIS-NN has the potential to enhance the efficiency of both neural network inference and image processing tasks. This optimization could result in accelerated pixel format conversion, ensuring memory efficiency and alignment with the limited resources of the microcontroller in the embedded system.

By amending the main.c that generated in STM32CubeIDE, we are able to perform the digit recognition by inserting the testing example into the model. By connecting the board to the laptop via USB cable which is USART.

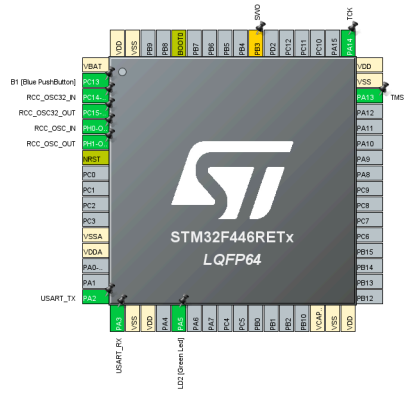

Figure 3: Pinout setting for STM32F446RE

Figure 3 has shown that we have set PA3 as USART_RX and PA2 and USART_TX. Besides,in this project, we use LED as our output to show what is the digit that has been recognised by blinking the LED, which is PA5.

STMCUBEMX.AI is used to generate C code from the trained model. The MNIST dataset in Keras model is inserted into the tools. By amending the code generated, we are able to recognize the example input digit that we inserted through blinking of LED.

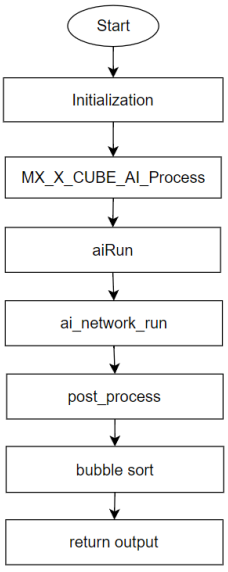

Figure 4: C code Function flow

Our system's algorithmic workflow is shown in Figure 4. Starting with initialization—which is produced by the IDE automatically in order to guarantee the best possible neural network performance—the procedure is carried out through the MX_X_CUBE_AI_Process function. This function acts as the entry point for the input data, which is defined in the data.h file and consists of a 28x28 array representing the digits 0 through 9. This function manages the input retrieval process, which pulls information from the previously stated header file.

The aiRun function then starts to operate. Batch characteristics and input and output data are obtained within this function. The weight parameters are multiplied by the input data using the ai_network_run function that follows. The output is then sent for post-processing.

In order to forecast the represented digit, the post_process function performs the vital work of comparing the output with the MNIST sample dataset. A dummy register is used to temporarily store data during this computation. After computation, the dummy output is provided to the prediction value, which is a crucial variable in the bubble sorting process that follows and is necessary to identify the most probable digit. The system output is this probability-sorted result, where the highest probability digit determines when the LED blinks. To give an example, the LED will blink twice if the input equals '2'.

V. RESULT AND DISCUSSION

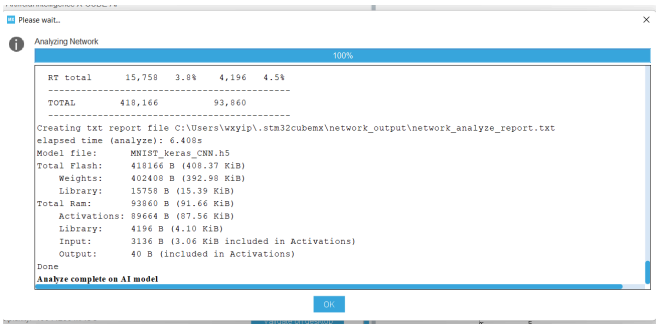

Figure 5: MNIST model analysis report

In order to build digit recognition system, MNIST dataset is used as the trained model in our project. By leveraging the pre-trained model, we have to train the model into our board. Before getting to train the model, we have to analyze the model to ensure the board has enough flash to support the model. As shown in Figure 5, our board is having sufficient flash to support the model.

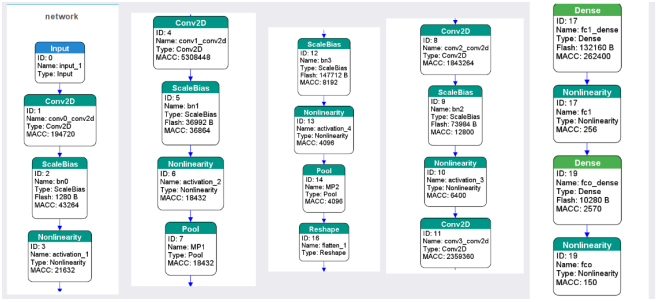

Figure 6: Block diagram of the network model generated through STMCubeAI

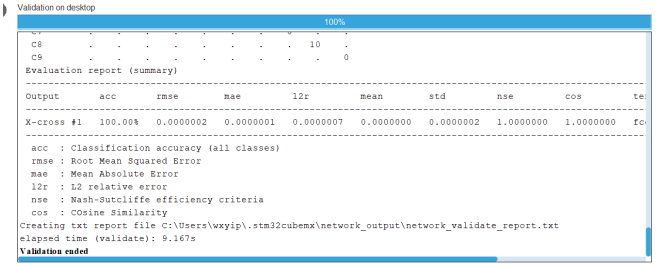

Figure 7: Validation report

By connecting the board to the laptop, we have validate the model to ensure the model is able to use in the board. Next we can generate the code template for further performance.

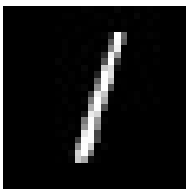

Figure 8: Digit ‘1’ test sample

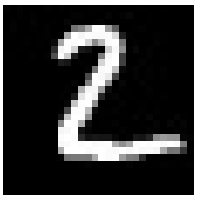

Figure 9: Digit ‘2’ test sample

Figure 8 and Figure 9 are the example test sample that we used in the system. After the calculation and prediction needed, the result will be shown through the LED (PA5). When Figure 8 is used as input, the LED is on after the simulation. With Figure 9 as input, the LED will blink twice after the simulation.

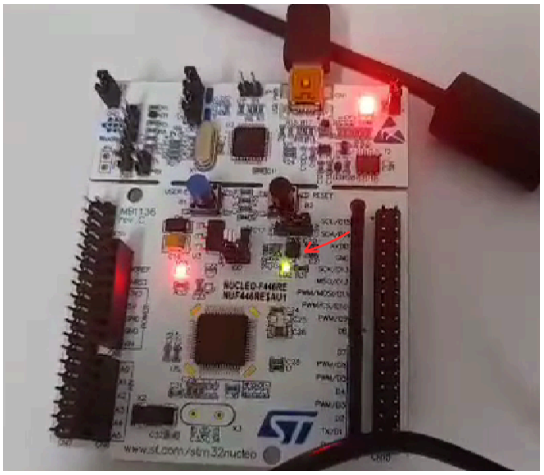

Figure 10: LED is on after ‘1’ is used as input

VI. CONCLUSION

We have successfully shown the functionality of on-device digit recognition using CMSIS-NN on the Discovery STM32nucleo-f446re board and TensorFlow Lite for Microcontrollers to wrap up this project. Using LED blinking to indicate recognized digits is one practical way that machine learning can be implemented in resource-constrained contexts.

Despite encountering a limitation with the printf function due to the UART error that we faced, which currently hinders the direct display of recognized digits, this project lays a solid foundation for future improvements. Addressing this issue and exploring alternative methods for visualizing the recognized digits will enhance the project's practical utility and user experience. With the improvement of visualization, we are able to provide a comprehensive evaluation of the system, future efforts will be directed towards implementing a mechanism to display accuracy metrics.

VII. REFERENCE

[1] Ioan Lucan Orășan and Cătălin Daniel Căleanu, “ARM Embedded Low Cost Solution for Implementing Deep Learning Paradigms,” Nov. 2020, doi: https://doi.org/10.1109/isetc50328.2020.9301130.

[2] u2man, “CIFAR10_STM32_Image_Classificatio/dataset/cifar10 model.h5 at

master . u2man/CIFAR10_STM32_Image_Classificatio,” GitHub, 2020.

https://github.com/u2man/CIFAR10_STM32_Image_Classificatio/blob/master/dataset/cifar1

0%20model.h5.

[3] “MNIST Dataset,” www.kaggle.com. https://www.kaggle.com/datasets/hojjatk/mnist-dataset

[4] Nima, “nimaaghli/STM32AI_MNIST,” GitHub, Aug. 20, 2023.

https://github.com/nimaaghli/STM32AI_MNIST.