Convert and optimize NN for MCU (MNIST)

Omar El Nahhas, 214316IV

I. Introduction

The goal of this assignment is to implement the model from home assignment 2 on the STM32F4, which is a deep-learning model optimised with respect to its memory size. The model is trained on the MNIST data set, containing 60000 training images and 10000 validation images (28x28; grayscale). The LCD screen, touchscreen and buttons on the STM32F4 are used to create new handwritten images to run inference on, with the aim to have on-demand, high accuracy classification of new input images.

II. Methodology

To start off, the LCD, touchscreen and buttons are configured in C to enable user interaction. Using the blue STM32F4 board, the y-coordinates of the screen are flipped so that the user interface and underlying coordinate system is correctly oriented. Afterwards, the model from home assignment 2 is uploaded on the board using CubeMX. The model was already converted to a .tflite model as it was the determined file type used to compare sizes between architectures. The neural network architecture of the model is presented in Fig. 1, with details of the model shown in Table I. As the model was optimised based on size, no quantization is necessary for the model to be loaded onto the board as it is far within capacity of the STM32F4, see Fig. 2. The validation accuracy is on the 10.000 MNIST images, and the test accuracy is on the 10 self-written samples created in home assignment 2.

The user interface is configured to detect touch and draws points on the big screen, which simultaneously draws an image in a 28x28 square in the right-bottom of the screen. This is because the model is trained on 28x28 images, and is therefore the required format of the input images as well. If the input size of the architecture would have been a different shape, the drawing box should be changed accordingly. When a figure is drawn, the user can either choose to clear the screen by pressing the on-screen clear button, or to perform inference by clicking on the on-board user button when satisfied with the drawn figure. Because an analogous button is used to initiate inference, the bouncing effect can cause multiple inference requests to be launched. It was considered building a debounce filter to deal with this phenomenon, but simply adding a small delay after the first trigger of a couple 100 ms already fixed this bouncing button problem. After initiating inference through the user button, the LCD screen shows the input figure, prediction and confidence altogether.

TABLE I: Information about the model.

Architecture	Params.	Size	Valid.	Test
See Fig. 1	707	6.5 KB	0.9636	1.0

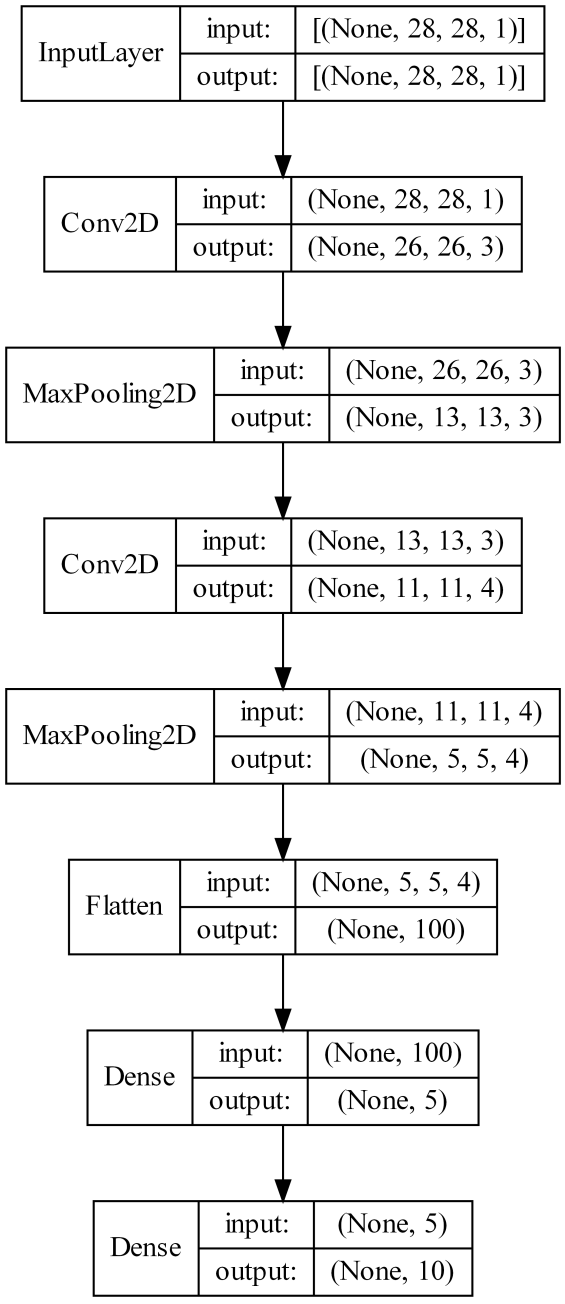

Fig. 1: The model used to run inference on the board.

III. Results

In order to test the accuracy of the model on the board, the single numbers 0-9 are drawn (in random order) on the STM32F4 to run inference on. The drawn numbers, the prediction and the confidence are shown in Fig. 3 and textually summarized in Table II.

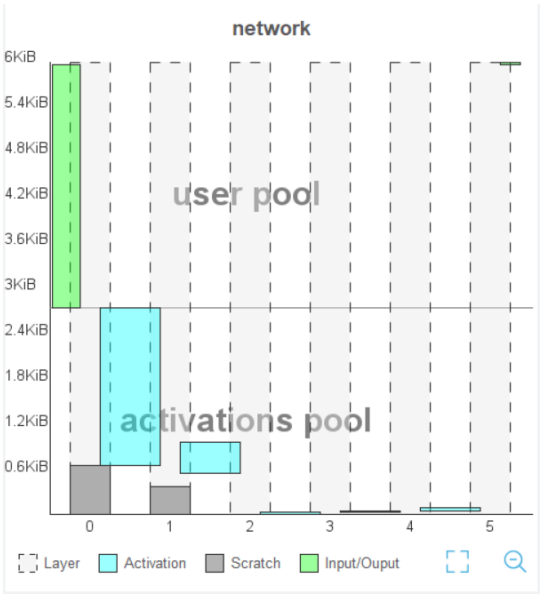

Fig. 2: Model memory occupation shown in CubeMX.

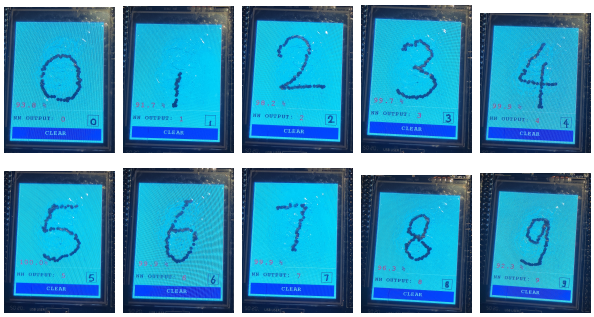

Fig. 3: Running inference on the model on the STM32F4.

TABLE II: Results of running inference on the board with self-written images.

Ground-truth	Prediction	Confidence
0	0	93.8%
1	1	91.7%
2	2	92.8%
3	3	99.7%
4	4	99.9%
5	5	100%
6	6	99.9%
7	7	99.9%
8	8	96.3%
9	9	92.3%

Observing the results, it can be concluded that the model is classifying the numbers accurately with high confidence. Despite the model being only 6.5 KB, it classifies every input number within the scope of this experiment correctly with a confidence higher than 90%.

IV. Discussion

The major issues which occurred during this home assignment were related to incompatible versions across software, hardware and firmware. This made debugging extremely difficult, as it was initially unclear if I was doing something wrong, or that something was wrong with the version of the (software) components I used. However, after support from the forums the problems were quickly resolved and I could focus on the actual implementation. The machine learning part of this assignment was quite straight-forward, as the model from home assignment 2 was already performing with desired accuracy and did not require any changes to make it fit.