LAB-10 REPORT

Subject: **Embedded Hardware Design**

Subject Code: **EL203**

Members: 1) Shantanu Tyagi **(201801015)**
2) Nikhil Mehta **(201801030)**
3) Sudhanshu Mishra **(201801114)**
4) Bhavana Kolli **(201701082)**

Introduction

Embedded artificial intelligence refers to the use of machines and deep learning in software at the device level (AI). The programme may be designed to give both predictive and reactive intelligence based on the data collected and reviewed. AI models may be run on devices having embedded AI, and the results can subsequently be used to achieve a purpose or function. TinyML (embedded machine learning) is a machine learning technique that is applied to embedded devices. Some of its benefits, when installed on systems, are listed below:

1) Less/No Bandwidth constraints
2) Improved latency
3) Cost-efficient

Standard computer programming addresses issues by expressing strategies directly in code and instructing computers to execute logic to process data and generate an output. Machine Learning (ML), on the other hand, is an AI technique for detecting patterns in data and, in essence, learning from it. Pre-labelling data (or not), using reinforcement learning to guide algorithm development, extracting features via statistical analysis, and then classifying input data against this trained data set to determine an output with a stated degree of confidence are all examples of how this can be accomplished.

In an ideal embedded Machine learning ecosystem, device providers, original equipment manufacturers (OEMs), and machine learning models are employed and deployed. It also expands the Machine Learning ecosystems by including approaches and tools for developing and deploying embedded devices, such as the Internet of Things (IoT), sometimes known as Artificial Intelligence on IoT devices (AIoT).

CODE:

uart.h file:

```
#ifndef _UART_H
#define _UART_H
#include "stm32f4xx.h"
#include <stdio.h>

void USART_Init(void);
void test_setup(void);

#endif
```

uart.c file:

```
#include "uart.h"

void USART2_Init (void) {
//Enable clock for UART. at APB1
	/*ENABLE clock for USART2*/
	RCC->APB1ENR |=  0x20000;
	RCC->AHB1ENR |=  0x1;

	/* Configure PA2, PA3 for USART2 TX, RX */
	GPIOA->AFR[0] &= ~0xFF00;
	GPIOA->AFR[0] |=  0x7700;   /* alt7 for USART2 */

	GPIOA->MODER  &= ~0x00F0;
	GPIOA->MODER  |=  0x00A0;   /* enable alt. function for PA2, PA3 */

	USART2->BRR = 0x0683;      /* 9600 baud @ 16 MHz */
	USART2->CR1 = 0x000C;      /* enable Tx, Rx, 8-bit data */
	USART2->CR2 = 0x0000;      /* 1 stop bit */
	USART2->CR3 = 0x0000;      /* no flow control */
	USART2->CR1 |= 0x2000;     /* enable USART2 */
}

int USART2_write(int ch) {
	while(!(USART2->SR & 0x0080)) {}
	USART2->DR =(ch &0xFF);
	return ch;
}

int USART2_read(void) {
	while(!(USART2->SR & 0x0020)) {}
	return USART2->DR;
```

```
}

struct __FILE {
	int handle;
};

FILE __stdin  = {0};
FILE __stdout = {1};
FILE __stderr = {2};

int fgetc(FILE *f) {
	int c;

	c = USART2_read();		/* read the character from console */

	if (c == '\r') {		/* if '\r', after it is echoed, a '\n' is
appended*/
		USART2_write(c);	/* echo */
		 c = '\n';
	}

    USART2_write(c);	/* echo */

	return c;
}

/* Called by C library console/file output */
int fputc(int c, FILE *f) {
	return USART2_write(c);  /* write the character to console */
}


int n;
char str[80];

void test_setup(void) {
	printf("Please enter a number:");
	scanf("%d", &n);
	printf("The number you entered is : %d\r\n", n);
	printf("Please enter a character string :");
	gets(str);
	printf("The string you entered is : ");
	puts(str);
	printf("\r\n");
}
```

simple_neural_networks.h file:

```
#ifndef __SIMPLE_NEURAL_NETWORK
#define __SIMPLE_NEURAL_NETWORK
```

```
#include <stdint.h>

double single_in_single_out (double input, double weight);
double multiple_inputs_single_out (double *input,
  double *weight,
  uint32_t INPUT_LEN);
void single_input_multiple_output_nn (double input_scalar,
double *weight_vector,
double *output_vector,
double VECTOR_LEN);

#endif
```

simple_neural_networks.c file:

```
int32_t single_in_single_out (int32_t input, int32_t weight)
{
       return (input * weight);
}

double weighted_sum (double *input, double *weight, uint32_t INPUT_LEN)
{
       double output;
       int i;
       for(int i = 0; i < INPUT_LEN; i++)
       {
       output += input[i] * weight[i];
       }
       return output;
}

double multiple_inputs_single_out (double *input, double *weight, uint32_t
INPUT_LEN)
{
       double predicted_data;
    predicted_data = weighted_sum(input, weight, INPUT_LEN);
       return predicted_data;

}

void simple_multiple (double input_scalar, double *weight_vector, double
*output_vector, double VECTOR_LEN) {
       int i;
       for (i = 0; i < VECTOR_LEN; i++) {
         output_vector[i] = input_scalar * weight_vector[i];
       }
}

void single_input_multiple_outputs_nn (double input_scalar,
double *weight_vector,
```

```
double *output_vector,
double VECTOR_LEN) {
    simple_multiple(input_scalar, weight_scalar, output_vector, VECTOR_LEN);
}
```

main.c file (single input):

```
// Single input single output Neural Network model to predict whether a person
is
// affected by COVID-19 or Healthy, given a particular temperature.

#include "uart.h"
#include "simple_neural_networks.h"

#define Covid 190

#define TEMPERATURE_IDX   0
#define BLOOD_PRESSURE_IDX 1
#define BLOOD_SUGAR_IDX   2

#define OUT_LEN 3

double prediction_data[3];

double weight[3] = {0.51,0.7,0.8};

int main(void)
{
      USART2_Init();
      single_input_multiple_outputs_nn(Covid, weights,
prediction_data,OUT_LEN);
     printf("Prediction of temperature:
%f\r\n",prediction_data[TEMPERATURE_IDX]);
      printf("Prediction of blood pressure:
%f\r\n",prediction_data[BLOOD_PRESSURE_IDX]);
      printf("Prediction of blood sugar:
%f\r\n",prediction_data[BLOOD_SUGAR_IDX]);

  while (1)
  {

  }
}
```

main.c (multiple input):

```
// Multiple input single output Neural Network model to predict whether a person
is
// affected by COVID-19 or Healthy, given a particular temperature, blood
```

```
pressure, and blood sugar.

#include "uart.h"
#include "simple_neural_networks.h"

#define SICK_IDX 0
#define COVID_IDX   1
#define ACTIVE_IDX  2

#define OUT_LEN 3
#define IN_LEN  3

double prediction_data[OUT_LEN];

double weights[OUT_LEN][IN_LEN] = {   {0.25, 0.6, 0.7},    //sick
                                      {0.25, 0.8, 1.2},    //covid
                                      {0.18, 0.6, 0.7}};  //active

double input_vector[IN_LEN] = {36.5, 120, 130};
int main (void)
{
	USART2_Init();

	multiple_inputs_multiple_outputs_nn(input_vector, IN_LEN,
prerdiction_data), OUT_LEN, weights);
printf("Prediction of sick: %f\r\n", prediction_data[SICK_IDX]);
printf("Prediction of Covid-19: %f\r\n", prediction_data[COVID_IDX]);
printf("Prediction of Active: %f\r\n", prediction_data[ACTIVE_IDX]);
	while(1)
	{

	}
}
```

simple_neural_networks.c file(with hidden layer):

```
int32_t single_in_single_out (int32_t input, int32_t weight)
{
	return (input * weight);
}

double weighted_sum (double *input, double *weight, uint32_t INPUT_LEN)
{
	double output;
	int i;
	for(int i = 0; i < INPUT_LEN; i++)
	{
	output += input[i] * weight[i];
	}
	return output;
```

```
}

double multiple_inputs_single_out (double *input, double *weight, uint32_t
INPUT_LEN)
{
	double predicted_data;
	predicted_data = weighted_sum(input, weight, INPUT_LEN);
	return predicted_data;

}

void simple_multiple (double input_scalar, double *weight_vector, double
*output_vector, double VECTOR_LEN) {
	int i;
	for (i = 0; i < VECTOR_LEN; i++) {
		output_vector[i] = input_scalar * weight_vector[i];
	}
}

void single_input_multiple_outputs_nn (double input_scalar,
double *weight_vector,
double *output_vector,
double VECTOR_LEN) {
	simple_multiple(input_scalar, weight_scalar, output_vector, VECTOR_LEN);
}

void matrix_multiplication (double *input_vector,
uint32_t INPUT_LEN,
double *output_vector,
uint32_t OUTPUT_LEN,
double weight_matrix[OUTPUT_LEN][INPUT_LEN]) {
	int k;
	int i;
	for (k = 0; k < OUTPUT_LEN; k++) {
		for (i = 0; i < INPUT_LEN; i++) {
		output_vector[k] += input_vector[i] * weight_matrix[k][i];
}
	}
}

void multiple_inputs_multiple_outputs_nn(double *input_vector,
uint32_t INPUT_LEN,
double *output_vector, uint32_t OUTPUT_LEN,
double  weight_matrix[OUTPUT_LEN][INPUT_LEN]){
	matrix_multiplication(input_vector, INPUT_LEN, output_vector, OUTPUT_LEN,
weight_matrix);
}

void hidden_nn(double *input_vector,
uint32_t INPUT_LEN,
uint32_t HIDDEN_LEN,
```

```
double in_to_hid_weight[HIDDEN_LEN][INPUT_LEN],
uint32_t OUT_LEN,
double hid_to_out_weight[OUT_LEN][HIDDEN_LEN],
double *output_vector) {
	double hidden_data_vector[HIDDEN_LEN];
	matrix_multiplication(input_vector, INPUT_LEN, hidden_data_vector,
OUT_LEN, in_to_hid_weight);
	matrix_multiplication(hidden_data_vector, HIDDEN_LEN, output_vector,
OUT_LEN, hid_to_out_weight);
}
```

main.c (multiple input and hidden layer):

```
// Multiple input single output Neural Network model to predict whether a person
is
// affected by COVID-19 or Healthy, given a particular temperature, blood
pressure, and blood sugar.

#include "uart.h"
#include "simple_neural_networks.h"

#define SICK_IDX 0
#define COVID_IDX   1
#define ACTIVE_IDX  2

#define OUT_LEN 3
#define IN_LEN  3
#define HID_LEN 3

double prediction_data[OUT_LEN];
//temp press sugar
double in_to_hid_weights[HID_LEN][IN_LEN] = {
{0.25, 0.6, 0.7},//HID[0]
	{0.25, 0.8, 1.2},//HID[1]
	{0.18, 0.6, 0.7} //HID[2]
};
//HID[0] HID[1] HID[2]
double hidden_to_out_weights[HID_LEN][IN_LEN] = {
{0.25, 0.6, 0.7},//Sick
	{0.25, 0.8, 1.2},//Covid-19
	{0.18, 0.6, 0.7} //Active
};

//temp press sugar
double input_vector[IN_LEN] = {36.5, 120, 130};

int main(void)
{
	USART2_Init();
```

```
      hidden_nn(input_vector, IN_LEN, HID_LEN, in_to_hid_weights,
OUT_LEN,hidden_to_out_weights,prediction_data);
printf("Prediction of sick: %f\r\n", prediction_data[SICK_IDX]);
printf("Prediction of Covid-19: %f\r\n", prediction_data[COVID_IDX]);
printf("Prediction of Active: %f\r\n", prediction_data[ACTIVE_IDX]);
      while(1)
      {

      }
}
```

Screenshot:

Conclusions/Observations:

- Machine learning may be used in a variety of fields, such as anticipating stock market collapses, ups and downs, and so on. We utilised machine learning to predict Covid and numerous biological things, as well as whether or not a person had Covid. We used a variety of strategies to anticipate the outcome.
- Unfortunately, we were unable to continue the experiment owing to hardware restrictions. However, using the simulation and code, we were able to gain a better understanding of the process and how it was implemented using the uart driver.
- In general, if we have previous data for anything, we can feed it to the computer, which will forecast the estimated outcome for that thing.