Integration of MATLAB-LSTM Model into foxBMS Firmware and Hex File Generation

---

✅Overview

This document outlines all the key modifications and steps needed to successfully integrate a MATLAB-trained LSTM model into the foxBMS-1 firmware, compile it, and generate a `.hex` file ready to be flashed onto an STM32F4 microcontroller.

✅Step-by-Step Summary

1. Prepare LSTM Model in MATLAB

- Train LSTM for SoC estimation.
- Ensure the model is configured to use `double` precision (default).
- Save the model as `trained_lstm.mat`.

▶ **Code Generation in MATLAB:**

```
cfg = coder.config('lib');
cfg.GenCodeOnly = true;
cfg.TargetLang = 'C';
cfg.Hardware = coder.Hardware('STM32F4xx Based');
% Not sure, maybe commented it
cfg.DeepLearningConfig = coder.DeepLearningConfig('TargetLibrary', 'none');
cfg.EnableOpenMP = false;           % Avoids omp.h errors
cfg.GenerateReport = true;

input_type = coder.typeof(0, [1, inf]);
codegen socEstimator.m -args {input_type, input_type, input_type} -config
cfg -report
```

---

2. Integrate Generated Code into foxBMS

A. Create New Folder

- Create a folder:

```
embedded-software/mcu-primary/src/application/sox/soc_lstm/
```

- Copy all generated `.c` / `.h` files from MATLAB's `codegen/lib/socEstimator/` to this folder.
- Exclude folders like `/examples` and `/html`

B. Remove `rtwtypes.h`

- Delete or ignore `rtwtypes.h` to avoid conflicts with foxBMS types.

C. Create `matlab_types_extension.h`

To supplement types expected by MATLAB code:

```c
#ifndef MATLAB_TYPES_EXTENSION_H_
#define MATLAB_TYPES_EXTENSION_H_

#include <stdint.h>
#include "matlab_types.h"

#ifndef real32_T
typedef float real32_T;
#endif

#ifndef real64_T
typedef double real64_T;
#endif

#ifndef int8_T
typedef int8_t int8_T;
#endif

#ifndef uint8_T
typedef uint8_t uint8_T;
#endif

#ifndef int16_T
typedef int16_t int16_T;
#endif

#ifndef uint16_T
typedef uint16_t uint16_T;
#endif

#ifndef uint32_T
typedef uint32_t uint32_T;
#endif
#endif
#ifndef boolean_T
typedef int boolean_T;
#endif
#ifndef true
#define true 1
#endif
#ifndef false
#define false 0
#endif
#ifndef MAX_int32_T
#define MAX_int32_T 2147483647
#endif
```

```
#ifndef MIN_int32_T
#define MIN_int32_T (-2147483647 - 1)
#endif

#endif
```

D. Update `sox.c` Files to Include the Above:

Include matlab types extension library:

```
#include "matlab_types_extension.h"
```

---

3. Write Wrapper Function

Create `socEstimator_wrapper.c`:

```
#include "socEstimator.h"
#include "socEstimator_initialize.h"
#include "socEstimator_terminate.h"
#include "socEstimator_emxAPI.h"
#include "socEstimator_emxutil.h"

void SOX_GetStateOfCharge_fromLSTM(double *voltage, double *current, double
*temperature, int len, double *soc_out) {
    static int is_initialized = 0;
    if (!is_initialized) {
        socEstimator_initialize();
        is_initialized = 1;
    }

    // Allocate arrays and call estimator
    // float soc_array[1024];  // adjust size as needed
    // float step_times[1024]; // optional, remove if unused

    emxArray_real_T *v_arr   = emxCreateWrapper_real_T(voltage, 1, len);
    emxArray_real_T *c_arr   = emxCreateWrapper_real_T(current, 1, len);
    emxArray_real_T *t_arr   = emxCreateWrapper_real_T(temperature, 1, len);
    emxArray_real_T *soc_arr = emxCreate_real_T(1, len);  // output

    socEstimator(v_arr, c_arr, t_arr, soc_arr);
    //socEstimator(voltage, current, temperature, soc_array);

    //*soc_out = soc_array[len - 1];  // take latest estimate
    *soc_out = soc_arr->data[len - 1];

    // Free arrays
    emxDestroyArray_real_T(v_arr);
    emxDestroyArray_real_T(c_arr);
    emxDestroyArray_real_T(t_arr);
    emxDestroyArray_real_T(soc_arr);
}
```

Declare in `socEstimator_wrapper.h`:

```
#ifndef SOC_ESTIMATOR_WRAPPER_H
#define SOC_ESTIMATOR_WRAPPER_H

void SOX_GetStateOfCharge_fromLSTM(double *voltage, double *current, double
*temperature, int len, double *soc_out);

#endif
```

---

4. Modify `` in foxBMS

- Replace or bypass Coulomb counting logic.

find SOC_Calculation function and replace it with following:

```
void SOC_Calculation(void) {

DB_ReadBlock(&sox_current_tab, DATA_BLOCK_ID_CURRENT_SENSOR);

DB_ReadBlock(&cellminmax, DATA_BLOCK_ID_MINMAX);\

float v = (float)(cellminmax.voltage_mean) / 1000.0f;     // mV to V

float c = (float)(sox_current_tab.current) / 1000.0f;     // mA to A

float t = (float)(cellminmax.temperature_mean);          // °C\

voltage_buffer[buffer_index] = v;

current_buffer[buffer_index] = c;

temperature_buffer[buffer_index] = t;

buffer_index++;

if (buffer_index >= LSTM_INPUT_LEN) {

buffer_index = 0;

buffer_full = true;
}

if (buffer_full) {

double soc_out = 0.0;

SOX_GetStateOfCharge_fromLSTM(voltage_buffer, current_buffer, temperature_buffer, LSTM_INPUT_LEN, &soc_out);

// Clamp the output

if (soc_out > 100.0) soc_out = 100.0;

if (soc_out < 0.0) soc_out = 0.0;

// Set SoC values

sox.soc_mean = soc_out;

sox.soc_min = soc_out;

sox.soc_max = soc_out;

// Store to NVM

SOX_SOC_s soc_struct = {soc_out, soc_out, soc_out, 0, 0, 0, 0};

NVM_setSOC(&soc_struct);

DB_WriteBlock(&sox, DATA_BLOCK_ID_SOX);

}

}
```

- Add include:

```
#include "socEstimator_wrapper.h"
```

- Ensure `double voltage_buffer[]`, `current_buffer[]`, and `temperature_buffer[]` are used.
- Replace `bool` with `boolean_T`, and use `true` / `false` from `matlab_types_extension.h`.
- Replace float literals `100.0f` with `100.0` (for double compatibility).

---

5. Update `wscript`

- Add all new `.c` files from `soc_lstm/` into the `srcs =` list of \ embedded-software/mcu-primary/src/application/wscript
- Example:

```
os.path.join('sox', 'soc_lstm', 'socEstimator.c'),
os.path.join('sox', 'soc_lstm', 'socEstimator_initialize.c'),
os.path.join('sox', 'soc_lstm', 'socEstimator_terminate.c'),
os.path.join('sox', 'soc_lstm', 'socEstimator_data.c'),
os.path.join('sox', 'soc_lstm', 'socEstimator_emxAPI.c'),
os.path.join('sox', 'soc_lstm', 'socEstimator_emxutil.c'),
os.path.join('sox', 'soc_lstm', 'predictAndUpdateState.c'),
os.path.join('sox', 'soc_lstm', 'resetState.c'),
os.path.join('sox', 'soc_lstm', 'socEstimator_wrapper.c'),
```

- Also add `'sox/soc_lstm'` to the `includes +=` section.

os.path.join('sox', 'soc_lstm'),

---

6. Build foxBMS Firmware

Run the following from the foxBMS *root* directory:

```
python tools/waf configure
python tools/waf build_primary
```

If all steps were followed correctly, you will see:

```
[279/279] Creating hex file ...
'build_primary' finished successfully
```

And in the ***\build\primary\embedded-software\mcu-primary\src\general\*** get:

```
foxbms_primary.hex
```

ready to flash using STM32CubeProgrammer.

Final Result

- A fully integrated LSTM model from MATLAB inside foxBMS
- Code that compiles cleanly
- A `.hex` file that is ready to flash on STM32