Motor Vibration Classification AIoT

The use case discussed in this document is to predict the DC motor vibration pattern classification with on chip capability of ARM processor and use of neural network on IoT edge device.

IoT Platfrom Selection:

The ARM platform used in this project is STM32F446 , the 32 bit MCU by ST Microelectronics having ARM Cortex M-4 architecture with Floating Point Unit (FPU) Capability. (https://www.st.com/en/microcontrollers-microprocessors/stm32f446.html#documentation)
The sensor selected 3 axis accelerometer ADXL345 by Analog Devices. (https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf)

End to End Deep Learning Pipeline Implementation:

As shown in figure below, the complete pipeline involves 6 stages. We will discuss each stage in detail one by one.

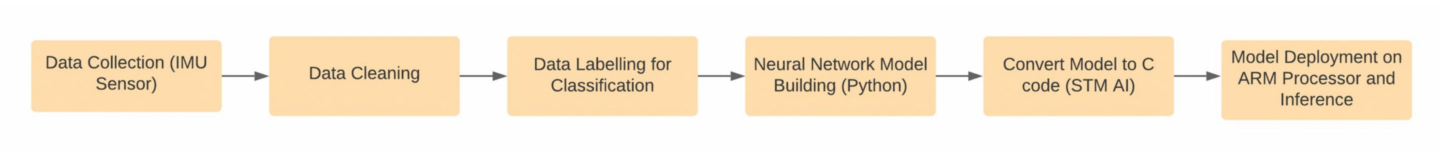

Data Collection (IMU Sensor)
Data Cleaning
Data Labelling for Classification
Neural Network Model Building (Python)
Convert Model to C code (STM AI)
Model Deployment on ARM Processor and Inference

Figure : Deep Learning Pipeline for on edge prediction

1. Data Collection :

The very basic step for implementation is to collect the necessary data from motor vibration, this needs proper selection of sensors and MCU since this is an IoT requirement. As discussed in introductory part the triaxial ADXL345 accelerometer and STM32F446 MCU were selected for data collection stage. Following figure shows the interfacing diagaram .As shown in the picture the ADXL345 and STM32F446 are being communicated via I2C interface and data is being transmitted to PC over UART interface through STM32F446. PC runs a python script to collect data over serial port and stores data in binary format. The handshake process takes as shown below

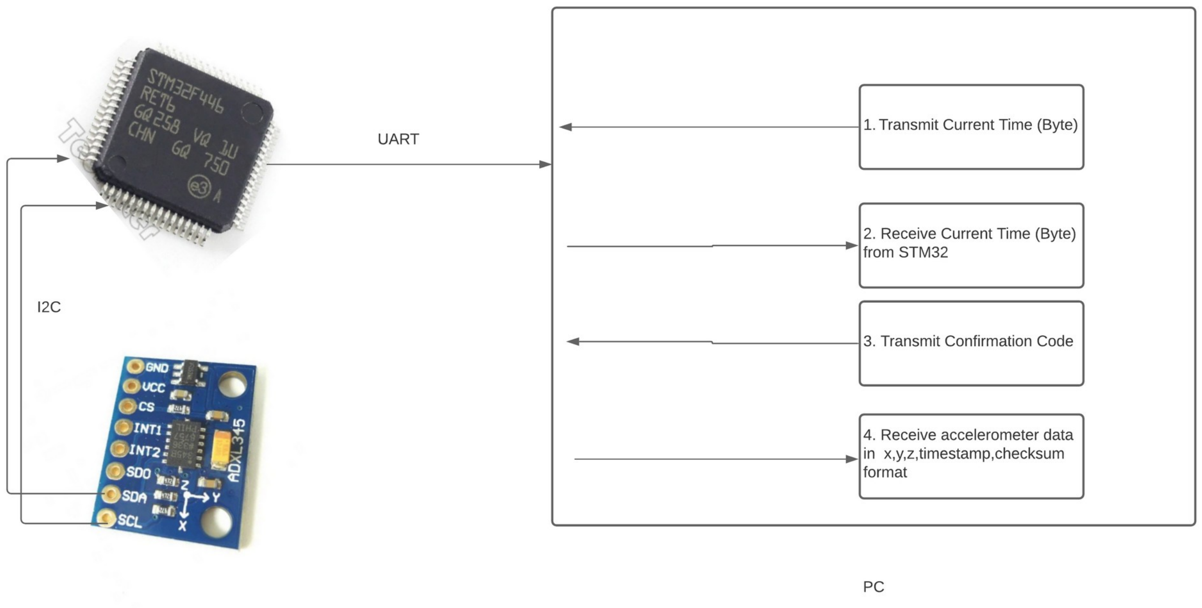

UART
I2C
1. Transmit Current Time (Byte)
2. Receive Current Time (Byte) from STM32
3. Transmit Confirmation Code
4. Receive accelerometer data in x,y,z,timestamp,checksum format
PC
STM32F446 RET6
ADXL345
GND
VCC
CS
INT1
INT2
SDO
SDA
SCL

Figure : Data collection and handshake process

2. Data Cleaning and Labeling :

The format in which data was collected is csv format with data represented as **x-axis,y-axis,z-axis,timestamp,checksum**. This format is necessary so as to have validated data from an IoT device. Since the data has come from motors and the scope of project is to classify the data based on the states of the motor viz. Normal, Idle,Fault1, Fault2, Fault3 and Fault4. Thus there will be 6 classes generated for data classification. In this stage data needs to be cleaned meaing null values or NaN values needs to be handled properly,also the existing format of data mentioned above needs to be updated as follows **machine id, pattern, timestamp, x-axis, y-axis, z-axis** .

3. Building a Neural network :

Considering a 2D data ,Convolutional Neural network has been chosen for prediction. As the final objective is to deploy model on STM32F446, the architecture of CNN is chosen such that it should fit in limited memory of MCU and also have accurate performance. The architecture of CNN for the problem statement is as shown in the next figure.

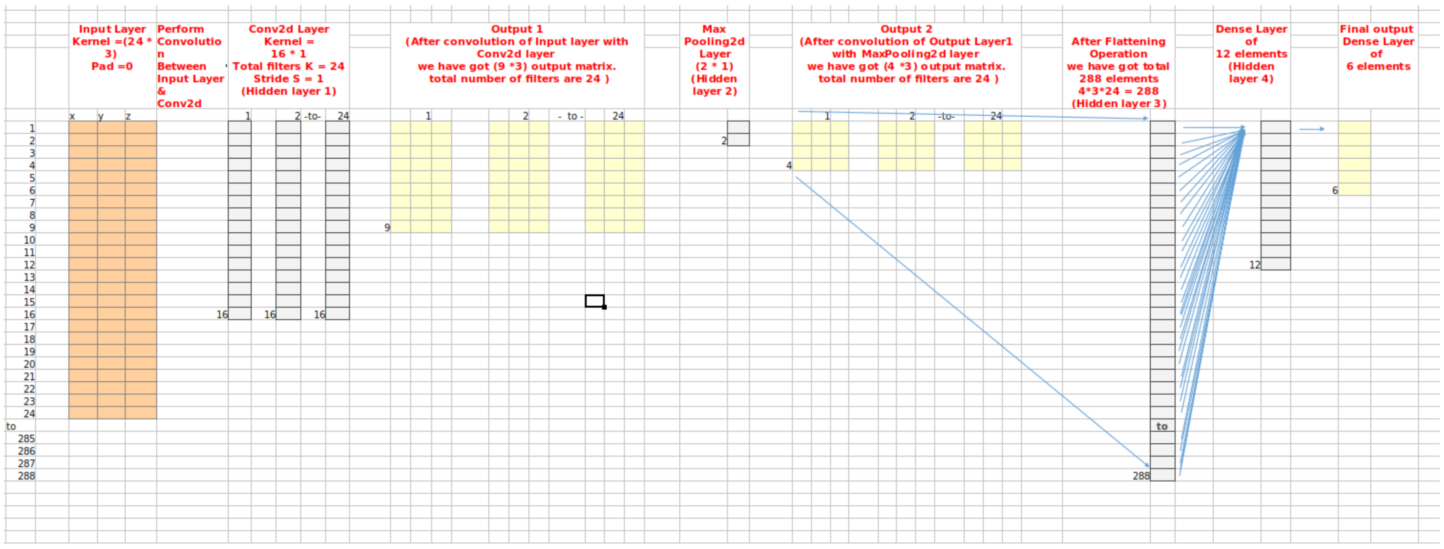
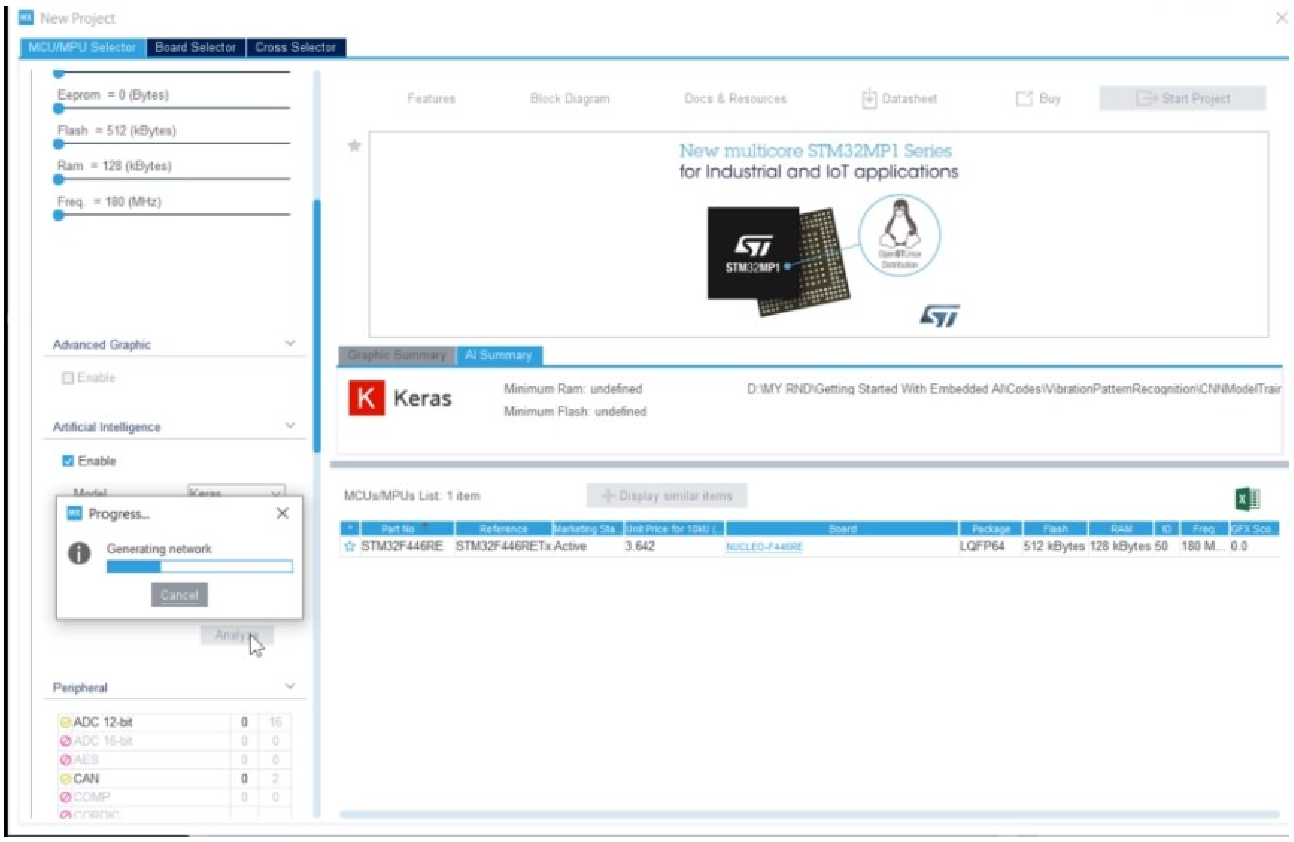

Input Layer Kernel =(24 * 3) Pad =0
Perform Convolution Between Input Layer & Conv2d
Conv2d Layer Kernel = 16 * 1 Total filters K = 24 Stride S = 1 (Hidden layer 1)
Output 1 (After convolution of Input layer with Conv2d layer we have got (9 *3) output matrix. total number of filters are 24)
Max Pooling2d Layer (2 * 1) (Hidden layer 2)
Output 2 (After convolution of Output Layer1 with MaxPooling2d layer we have got (4 *3) output matrix. total number of filters are 24)
After Flattening Operation we have got total 288 elements 4*3*24 = 288 (Hidden layer 3)
Dense Layer of 12 elements (Hidden layer 4)
Final output Dense Layer of 6 elements
x
y
z
1
2 -to-
24
- to -
-to-
16
9
2
4
to
285
286
287
288
12
6

Figure : CNN Architecture

The description of CNN is as follows,
It's basically a sequential model with one input ,output layer and four hidden layers of convolution 2D, max pooling 2D, flatten and dense layer.
Let's explore what happens inside this model of a neural network architecture while training and prediction. The first input layer of CNN is 24*3*1 i.e 24 rows, 3 columns and 1 channel. 24 rows are nothing but samples from accelerometer buffered from 3 columns i.e x,y and z axis of sensor. In the second layer i.e a convolution 2D layer of 16*1 convolves with the input layer with stride equal to one means when we will move the convolutional layer while converging with the input layer, it will move by a single element. The filter has been set to 24 to extract the features of input layer. Thus we get 24*3 matrices for the corresponding accelrometer channel. After convolving with hidden layer of max pooling, the next layer will have 24*3 matrices. Post to that the flattening applied on the output to vectorize the data in a single column. Thus total 288 elements will be present in the vector. For the fully connected layers dense layer of 12 elements was added. The final output will be dense layer of 6 elements as we have 6 classess for prediction. Once model is executed its performance will be evaluated from confusion matrix which is more than 90% in this case and for making it superior, the max pooling layer has been updated in the code.

4. Converting model to C code and Model Deployment :

To convert CNN model from keras in .h5 format to C code, we use ST-Cube AI utility which uses CMSIS-NN underneath. Following sequence of images shows how it converts the keras model to C code for STM32F446.



Eeprom = 0 (Bytes)
Flash = 512 (kBytes)
Ram = 128 (kBytes)
Freq. = 180 (MHz)

Enable (Advanced Graphic, unchecked)
Enable (Artificial Intelligence, checked)
Model: Keras

Progress...
Generating network

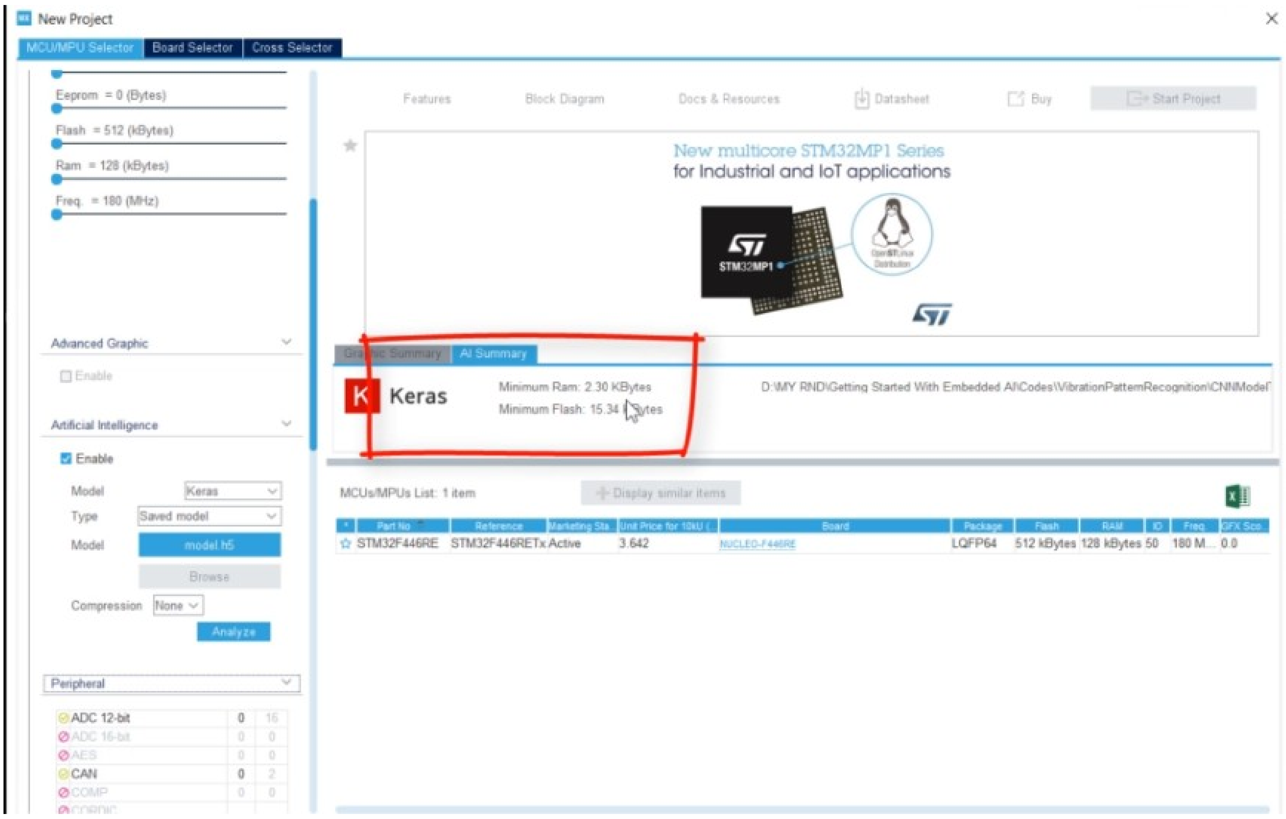
Keras
Minimum Ram: undefined
Minimum Flash: undefined
D:\MY RND\Getting Started With Embedded AI\Codes\VibrationPatternRecognition\CNNModelTrair

MCUs/MPUs List: 1 item

*	Part No	Reference	Marketing Sta...	Unit Price for 10kU (...	Board	Package	Flash	RAM	IO	Freq.	GFX Sco...
☆	STM32F446RE	STM32F446RETx	Active	3.642	NUCLEO-F446RE	LQFP64	512 kBytes	128 kBytes	50	180 M...	0.0

Peripheral		
ADC 12-bit	0	16
ADC 16-bit	0	0
AES	0	0
CAN	0	2
COMP	0	0

Figure : Conversion of Keras CNN model to C code via STcube AI



Eeprom = 0 (Bytes)
Flash = 512 (kBytes)
Ram = 128 (kBytes)
Freq. = 180 (MHz)

Enable (Artificial Intelligence, checked)
Model: Keras
Type: Saved model
Model: model.h5
Compression: None

Keras
Minimum Ram: 2.30 KBytes
Minimum Flash: 15.34 KBytes
D:\MY RND\Getting Started With Embedded AI\Codes\VibrationPatternRecognition\CNNModel

MCUs/MPUs List: 1 item

*	Part No	Reference	Marketing Sta...	Unit Price for 10kU (...	Board	Package	Flash	RAM	IO	Freq.	GFX Sco...
☆	STM32F446RE	STM32F446RETx	Active	3.642	NUCLEO-F446RE	LQFP64	512 kBytes	128 kBytes	50	180 M...	0.0

Peripheral		
ADC 12-bit	0	16
ADC 16-bit	0	0
AES	0	0
CAN	0	2
COMP	0	0

Figure : Flash and RAM required for CNN execution on STM32F446 MCU

Once the CNN C code is generated, next step is to integrate this code with existing sesnor interfacing code. Thus integrated code when converted to binary and uploaded to the STM32F446 is the model deployment on an MCU. One can then infer the results by connecting sensors over the motor.