Guide for Deploying your TensorFlow Model in Cortex M based Microcontroller

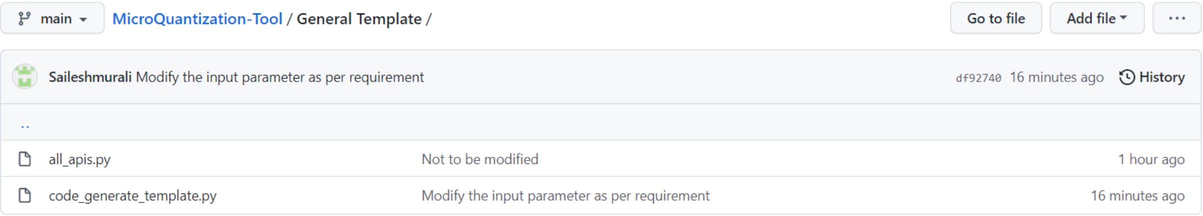
This manual tells a step-by-step procedure on how to use the codes given in this GitHub repository to automatically generate the inference code and all the related weights and biases in q7 format.

Two examples for quantization, one for Optical Character Recognition and another for prediction of sin(x) also have been given in the repository.

Step 1: Access the GitHub repository

https://github.com/kskumaree/MicroQuantization-Tool

Step 2: Open the General Template Folder

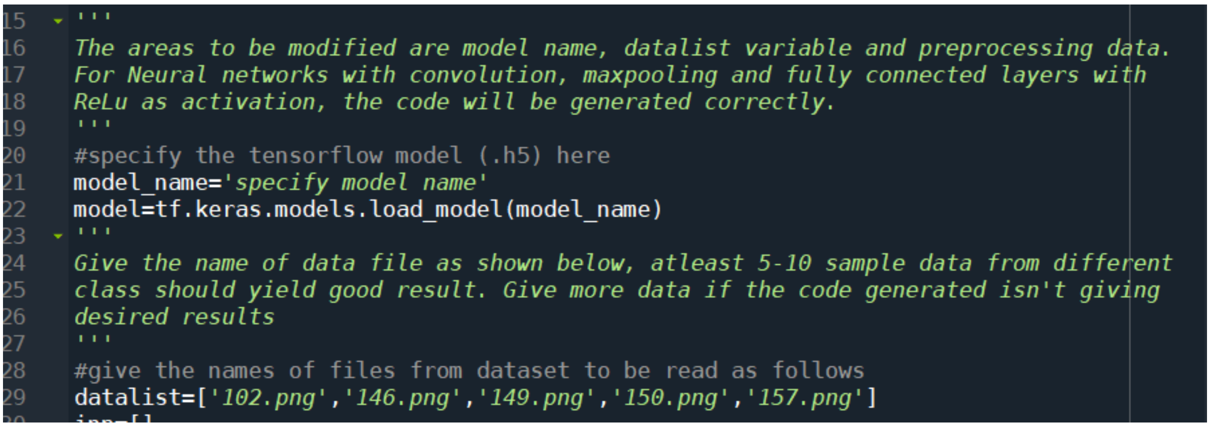


Saileshmurali Modify the input parameter as per requirement — df92740 16 minutes ago

all_apis.py	Not to be modified	1 hour ago
code_generate_template.py	Modify the input parameter as per requirement	16 minutes ago

You will find two codes, all_apis.py and code_generate_template.py there. Both of these, should be kept under the same path while executing. Note that all_apis.py file shouldn't be modified.

Step 3: Open the code_generate_templaye.py file



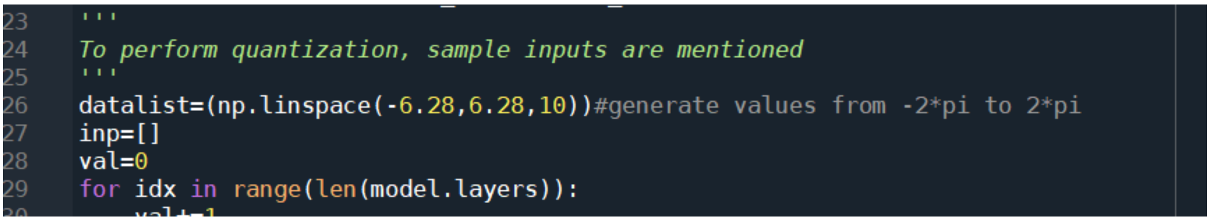
```
'''
The areas to be modified are model name, datalist variable and preprocessing data.
For Neural networks with convolution, maxpooling and fully connected layers with
ReLu as activation, the code will be generated correctly.
'''
#specify the tensorflow model (.h5) here
model_name='specify model name'
model=tf.keras.models.load_model(model_name)
'''
Give the name of data file as shown below, atleast 5-10 sample data from different
class should yield good result. Give more data if the code generated isn't giving
desired results
'''
#give the names of files from dataset to be read as follows
datalist=['102.png','146.png','149.png','150.png','157.png']
inp=[]
```

Here, specify the TensorFlow model name.

Step 4: Fill the input list

In the datalist variable, the input should be mentioned. Here, for example if we are reading images as the input, then image file name should be mentioned.

Another example for the time series model input is shown below.

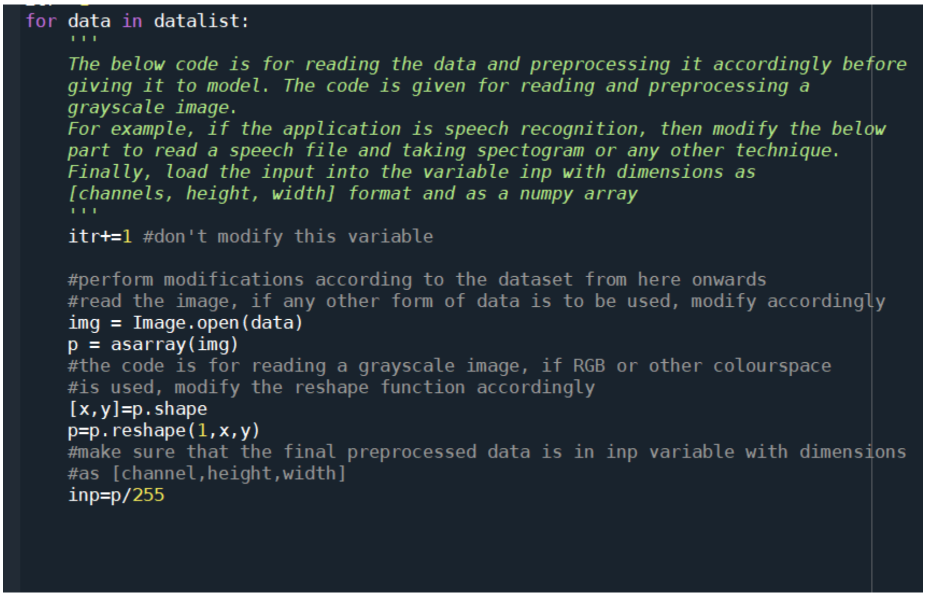


```
'''
To perform quantization, sample inputs are mentioned
'''
datalist=(np.linspace(-6.28,6.28,10))#generate values from -2*pi to 2*pi
inp=[]
val=0
for idx in range(len(model.layers)):
```

Note that it is essential to provide sample input ranging from maximum to minimum values in case of time series model.

In other cases, atleast 10 images/data from the training/validation set from different classes is to be given. More the input, better the result. The data can be from either training or validation set and it is important to give data from different classes. This part is crucial for quantizing the model.

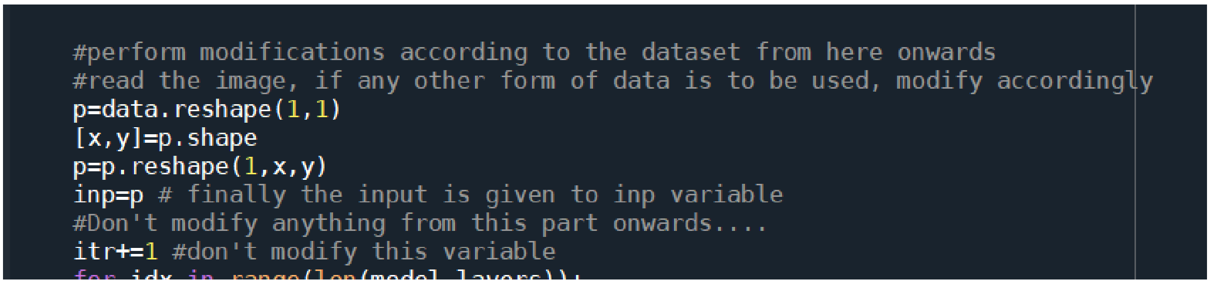
Step 5: Perform Pre-Processing



```
for data in datalist:
    '''
    The below code is for reading the data and preprocessing it accordingly before
    giving it to model. The code is given for reading and preprocessing a
    grayscale image.
    For example, if the application is speech recognition, then modify the below
    part to read a speech file and taking spectogram or any other technique.
    Finally, load the input into the variable inp with dimensions as
    [channels, height, width] format and as a numpy array
    '''
    itr+=1 #don't modify this variable

    #perform modifications according to the dataset from here onwards
    #read the image, if any other form of data is to be used, modify accordingly
    img = Image.open(data)
    p = asarray(img)
    #the code is for reading a grayscale image, if RGB or other colourspace
    #is used, modify the reshape function accordingly
    [x,y]=p.shape
    p=p.reshape(1,x,y)
    #make sure that the final preprocessed data is in inp variable with dimensions
    #as [channel,height,width]
    inp=p/255
```

As mentioned in above figure for OCR model, access the file from system, perform the pre-processing before feeding into the neural network. The pre-processing steps should be the same from that did during training. Steps are shown for the time series model below.

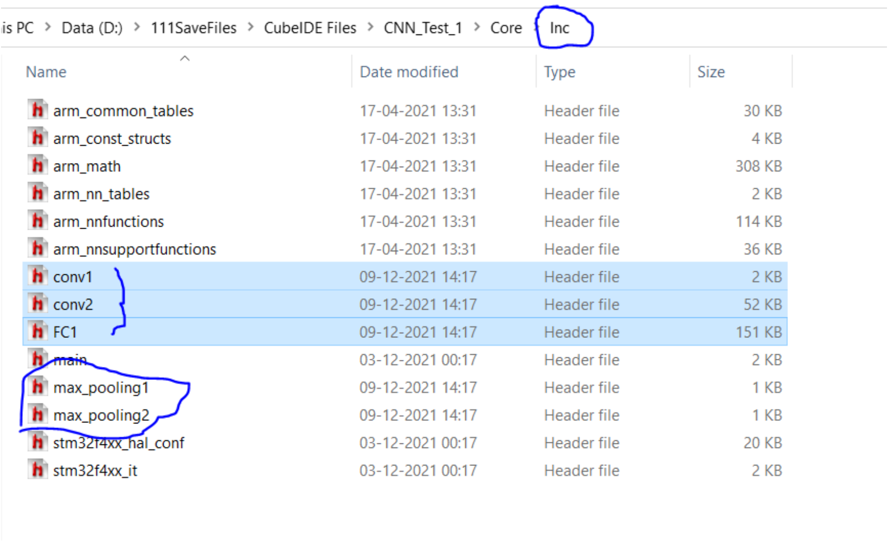


```
#perform modifications according to the dataset from here onwards
#read the image, if any other form of data is to be used, modify accordingly
p=data.reshape(1,1)
[x,y]=p.shape
p=p.reshape(1,x,y)
inp=p # finally the input is given to inp variable
#Don't modify anything from this part onwards....
itr+=1 #don't modify this variable
for idx in range(len(model.layers)):
```

Finally, save the input value to the variable “inp”. It should be a numpy array with dimensions as [channels, height, width] format.

Step 6: File Generation

After following till step 5, run the program and the files will be generated in the directory of the python script itself. Copy and paste the .h files generated and put it in the Inc folder as shown below.

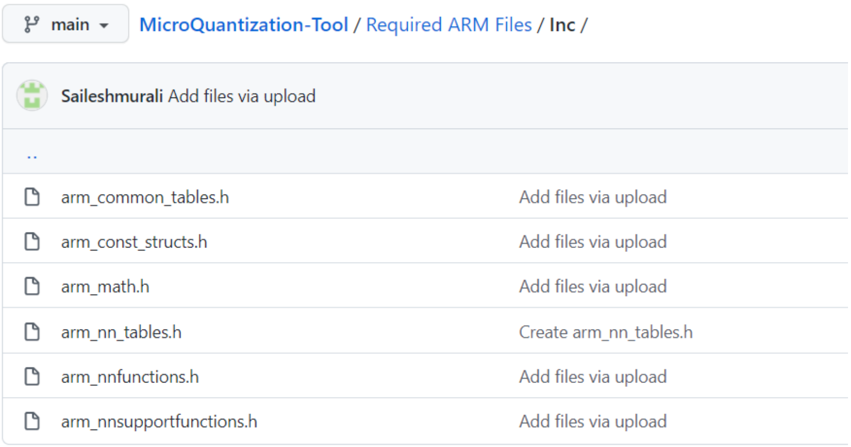


is PC > Data (D:) > 111SaveFiles > CubeIDE Files > CNN_Test_1 > Core > Inc

Name	Date modified	Type	Size
arm_common_tables	17-04-2021 13:31	Header file	30 KB
arm_const_structs	17-04-2021 13:31	Header file	4 KB
arm_math	17-04-2021 13:31	Header file	308 KB
arm_nn_tables	17-04-2021 13:31	Header file	2 KB
arm_nnfunctions	17-04-2021 13:31	Header file	114 KB
arm_nnsupportfunctions	17-04-2021 13:31	Header file	36 KB
conv1	09-12-2021 14:17	Header file	2 KB
conv2	09-12-2021 14:17	Header file	52 KB
FC1	09-12-2021 14:17	Header file	151 KB
main	03-12-2021 00:17	Header file	2 KB
max_pooling1	09-12-2021 14:17	Header file	1 KB
max_pooling2	09-12-2021 14:17	Header file	1 KB
stm32f4xx_hal_conf	03-12-2021 00:17	Header file	20 KB
stm32f4xx_it	03-12-2021 00:17	Header file	2 KB

Inference.c fill will be generated which contains the inference function. Paste it in the main.c file of your program and call the function to make the inference.

Also add the required .h and .c files which are needed to run the inference code. They are available in the repository under “Required Arm Files” folder.

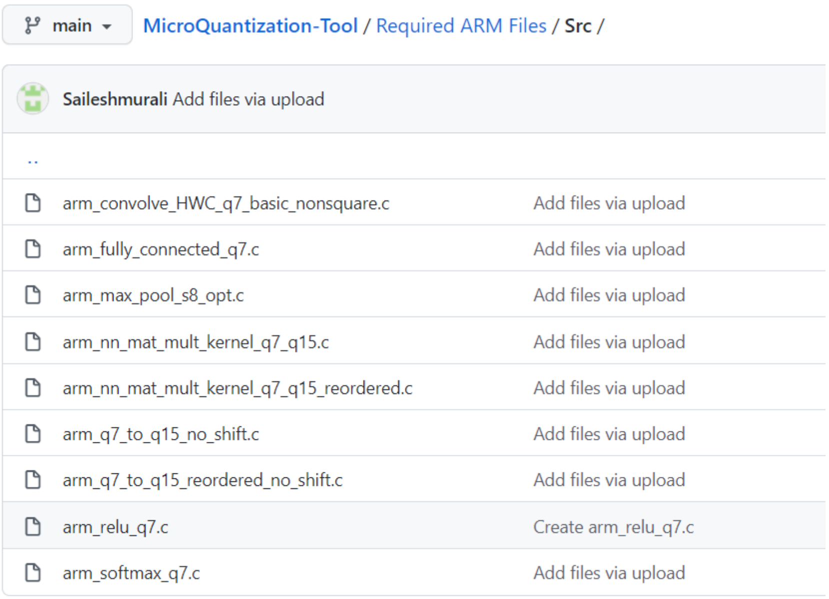


main / MicroQuantization-Tool / Required ARM Files / Inc /

Saileshmurali Add files via upload

..

arm_common_tables.h	Add files via upload
arm_const_structs.h	Add files via upload
arm_math.h	Add files via upload
arm_nn_tables.h	Create arm_nn_tables.h
arm_nnfunctions.h	Add files via upload
arm_nnsupportfunctions.h	Add files via upload



main / MicroQuantization-Tool / Required ARM Files / Src /

Saileshmurali Add files via upload

..

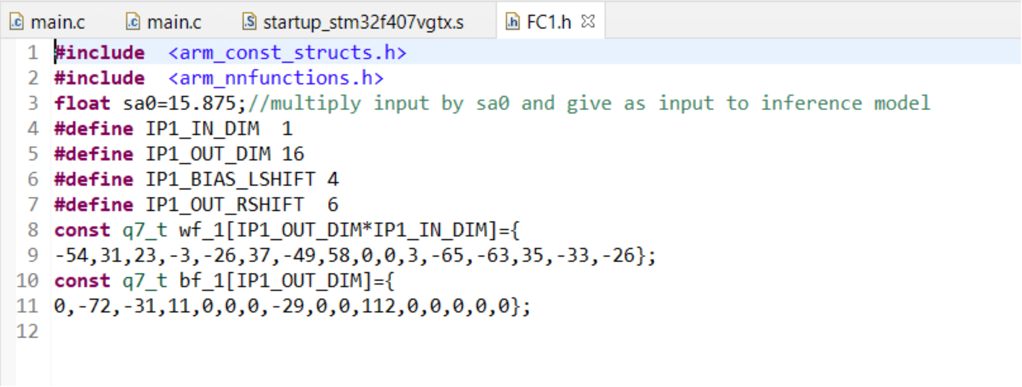
arm_convolve_HWC_q7_basic_nonsquare.c	Add files via upload
arm_fully_connected_q7.c	Add files via upload
arm_max_pool_s8_opt.c	Add files via upload
arm_nn_mat_mult_kernel_q7_q15.c	Add files via upload
arm_nn_mat_mult_kernel_q7_q15_reordered.c	Add files via upload
arm_q7_to_q15_no_shift.c	Add files via upload
arm_q7_to_q15_reordered_no_shift.c	Add files via upload
arm_relu_q7.c	Create arm_relu_q7.c
arm_softmax_q7.c	Add files via upload

Add all the files given in the repository. Header files should be added in Inc folder and .c files in Src folder.

All the steps mentioned are general and should be common for all IDEs. After doing all the mentioned things, you'll be ready to use the neural network in the Cortex M microcontroller.

Step 7: Using the inference code generated

The input to the inference engine in C should be in q7 format. In this step, we'll guide you on how to convert your input data which can be in floating point or other formats into q7. In the header files generated, there will be a variable sa0 declared in the first layer's corresponding file. For the sine model, since the first layer is Fully connected, it will be in FC1.h file and for OCR model, it will be in conv1.h model.

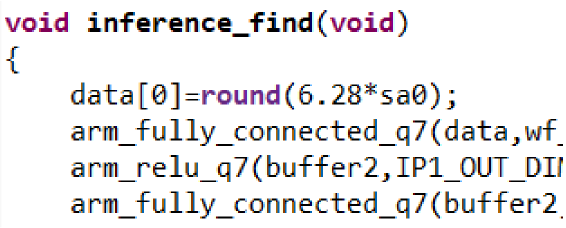


```
#include  <arm_const_structs.h>
#include  <arm_nnfunctions.h>
float sa0=15.875;//multiply input by sa0 and give as input to inference model
#define IP1_IN_DIM  1
#define IP1_OUT_DIM 16
#define IP1_BIAS_LSHIFT 4
#define IP1_OUT_RSHIFT  6
const q7_t wf_1[IP1_OUT_DIM*IP1_IN_DIM]={
-54,31,23,-3,-26,37,-49,58,0,0,3,-65,-63,35,-33,-26};
const q7_t bf_1[IP1_OUT_DIM]={
0,-72,-31,11,0,0,0,-29,0,0,112,0,0,0,0,0};

```

The variable can be seen in above screenshot.

To quantize the input, you just have to multiply the entire input array by the sa0 variable and round off the result. The screenshot below is for quantizing the sine model's input. If the size of input is more than 1, the entire array has to be multiplied and rounded off as shown below. Here, since the input size is 1, only one index is quantized. Quantizing the input is necessary to get correct result.



```
void inference_find(void)
{
    data[0]=round(6.28*sa0);
    arm_fully_connected_q7(data,wf_
    arm_relu_q7(buffer2,IP1_OUT_DI
    arm_fully_connected_q7(buffer2
```

Step 8: Understanding Variables

The variables used, their data types and what their size should be will be mentioned in the inference.c file generated in comments. Follow it accordingly.

The input variable is data and steps on how to quantize the input is mentioned in previous step. The output will be in SOFT_OUT variable if the last layer has softmax activation. Else, it will be in FC_OUT variable.

Conclusion

As mentioned before, for reference quantized model for sin(x) prediction and OCR prediction along with test images are given along with the python code to perform the automatic code generation and quantization. Also, a main.c with working and tested code are given both models. Feel free to test it and see the output. We hope that this proposed framework helps you to deploy neural networks in microcontroller easily.