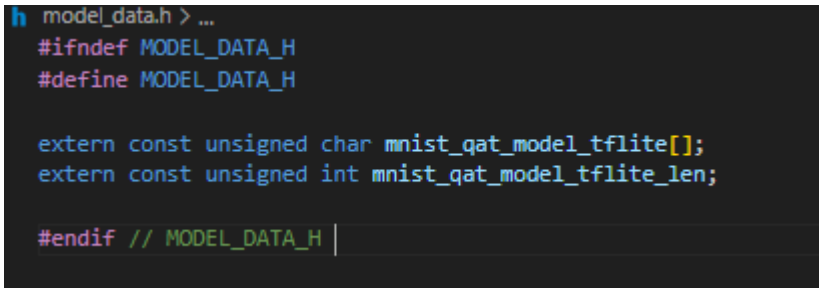
Steps to reproduce on **Windows**

WASM file generating

Model

Model generating code can be found in model_training, convert model to c array and place in model_c_array.cc + model_data.h

```
#include "model_data.h"

// We need to keep the data array aligned on some architectures.
#ifdef __has_attribute
#define HAVE_ATTRIBUTE(x) __has_attribute(x)
#else
#define HAVE_ATTRIBUTE(x) 0
#endif
#if HAVE_ATTRIBUTE(aligned) || (defined(__GNUC__) && !defined(__clang__))
#define DATA_ALIGN_ATTRIBUTE __attribute__((aligned(4)))
#else
#define DATA_ALIGN_ATTRIBUTE
#endif

const unsigned char mnist_qat_model_tflite[] DATA_ALIGN_ATTRIBUTE = {
  0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00,
  0x1c, 0x00, 0x18, 0x00, 0x14, 0x00, 0x10, 0x00, 0x0c, 0x00, 0x00, 0x00,
  0x08, 0x00, 0x04, 0x00, 0x14, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00,
  0x8c, 0x00, 0x00, 0x00, 0x0c, 0x01, 0x00, 0x00, 0xe4, 0x08, 0x00, 0x00,
  0xf4, 0x08, 0x00, 0x00, 0x50, 0x1f, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0a, 0x00,
```

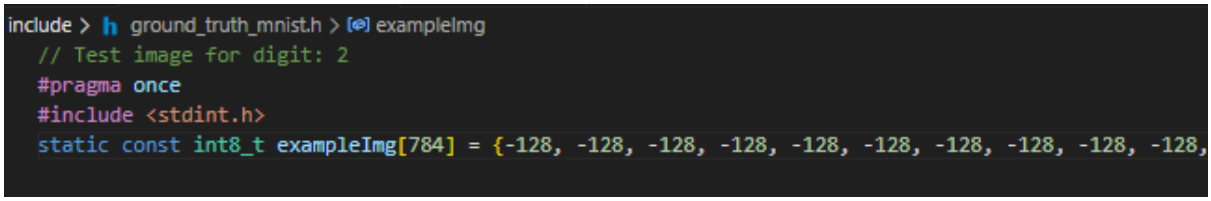


```
model_data.h > ...
#ifndef MODEL_DATA_H
#define MODEL_DATA_H

extern const unsigned char mnist_qat_model_tflite[];
extern const unsigned int mnist_qat_model_tflite_len;

#endif // MODEL_DATA_H
```

Ground image

To keep code and memory as clean as possible, the project uses images converted to c array. Script can be found inside /root/script: replace image url by new image url and let the script run

this will convert an image to the right properties, quantized etc and transform to c array, which can be imported in main.cpp



TFLite infer

Not much to change inside main.cpp, if you kept the image binary name (exampleImg), you can tweak tensorarena and operators used if the model has been changed. For the current MNIST model (+- 8kb), a tensorarena of 8kb is sufficient.

Building WASM file

On windows following command can be used when inside /root/src

!! Make sure to have the WASI SDK on your pc
Seth path variable: export WASI_SDK_PATH="C:/wasi-sdk-27.0-x86_64-windows"

Building: bash ./build.sh ← important, run this in git bash for example to be able to run shell scripts.

This will then generate test.wasm inside the src folder.

Run on windows

Inside /root/execute, I provided a built iwasm executable which will serve as WAMR host.

Make sure to place your wasm file (if created a new on) inside /compile

Then you can run the following command (on powershell):

inside execute folder:

./build/release/iwasm -v=5 --stack-size=8192 --heap-size=16384 --interp --stack-size=4096 --function tflite_infer ./compile/test.wasm

```
PS C:\Users\ceulea\final\locql\wamr_esp\execute> ./build/release/iwasm -v=5 --stack-size=8192 --heap-size=16384 --interp --stack-size=4096 --function tflite_infer ./compile/tes
[10:25:17:056 - 7FF7D4002500]: Load type section success.

[10:25:17:057 - 7FF7D4002500]: Load import section success.

[10:25:17:058 - 7FF7D4002500]: Load function section success.

[10:25:17:059 - 7FF7D4002500]: Load table section success.

[10:25:17:060 - 7FF7D4002500]: Load memory section success.

[10:25:17:060 - 7FF7D4002500]: Load global section success.

[10:25:17:060 - 7FF7D4002500]: Load export section success.

[10:25:17:060 - 7FF7D4002500]: Load table segment section success.

[10:25:17:061 - 7FF7D4002500]: Load datacount section success.

[10:25:17:061 - 7FF7D4002500]: Load code segment section success.

[10:25:17:061 - 7FF7D4002500]: Load data segment section success.

[10:25:17:061 - 7FF7D4002500]: Ignore custom section [target_features].
[10:25:17:062 - 7FF7D4002500]: Found aux __heap_base global, value: 34544
[10:25:17:062 - 7FF7D4002500]: Found aux __data_end global, value: 34532
[10:25:17:078 - 7FF7D4002500]: Found aux stack top global, value: 4096, global index: 0, stack size: 4096
[10:25:17:085 - 7FF7D4002500]: Load module success.

[10:25:17:086 - 7FF7D4002500]: Memory instantiate:
[10:25:17:086 - 7FF7D4002500]:   page bytes: 81920, init pages: 1, max pages: 1
[10:25:17:087 - 7FF7D4002500]:   heap offset: 65536, heap size: 16384

[10:25:17:087 - 7FF7D4002500]: Memory instantiate success.
enter tflite_infer function
getting model
building OP resolver
constructing interpreter
allocating (arena=8192)
Loading test image
invoking interpreter
done, digit=2
0x0:i32
PS C:\Users\ceulea\final\locql\wamr_esp\execute>
```