3rd July 2020

Handwritten Digit Recognition using Neural Network on Tiva Board

E3-257 Embedded Systems 2020

Course Project

By,

Naveen Chander (17550) and Sai Vamshi (17083),

DESE,

Indian Institute of Science

Project Goal

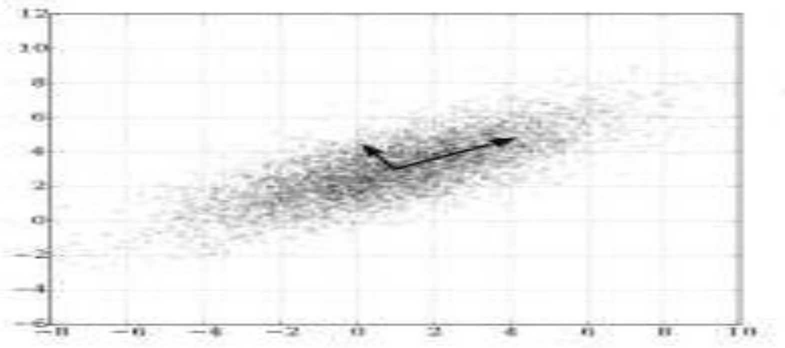
- Study the Handwritten Digits Recognition from Machine Learning perspective
- Define a Neural Network model and train the model
- Compress the model without significantly compromising on its accuracy
- Implement the obtained neural network on Tiva board by building a wrapper for user interface.
- Test and evaluation

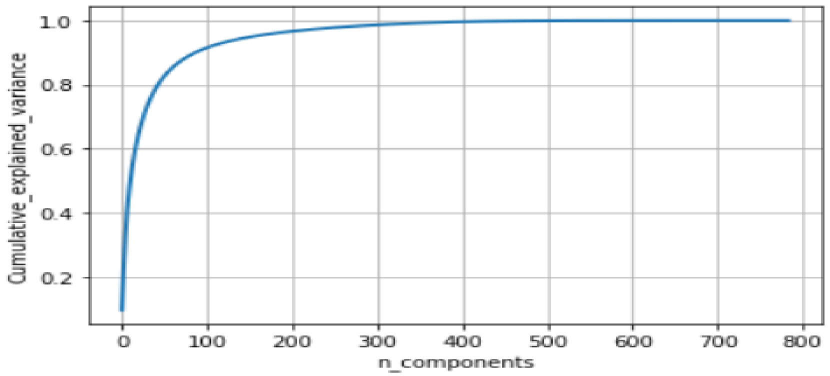
Data Handling

- The training set has 60K 28*28 images and test set has a 10K 28*28 images.
- The value of each pixel is in the range of 0-255 which is normalized between 0-1.
- The flattened image has a 784 features and if each feature uses 1 bytes then it requires almost 0.8KB just for the input.
- Therefore first the dataset is minimized using PCA(Principal Component Analysis.)

PCA

- This is nothing but projection of data onto lower dimension subspace so that maximum information is retained.
- Involves finding data covariance matrix its eigen values and eigen vectors.
- Eigen vectors are used as the axes onto which the data is projected.
- In our case we have used 100 features which retains around 92% of variance with 85% reduction in number of parameters.





Training on TensorFlow

- TensorFlow python package is used for training the network.
- First the network architecture must be fixed before implementing on the hardware (#parameters,accuracy and hardware constraints must be taken into account .)
- The best network is found by the trying various architectures and using a index which measures how much a network parameter contributes to the accuracy.
- The best network found has 24160 parameters which gives an accuracy of 97.5% and the architecture involves four layers with nodes [100 30 16 10].The activation function used is ReLu.
- The algorithm used for training is backpropagation.

Quantisation

- During training all the numbers used were floating point numbers.
- For the implementation in hardware we have done a quantisation analysis which says how accuracies vary for various number of bits used.
- Results of quantisation in next slide.

Quantisation Results

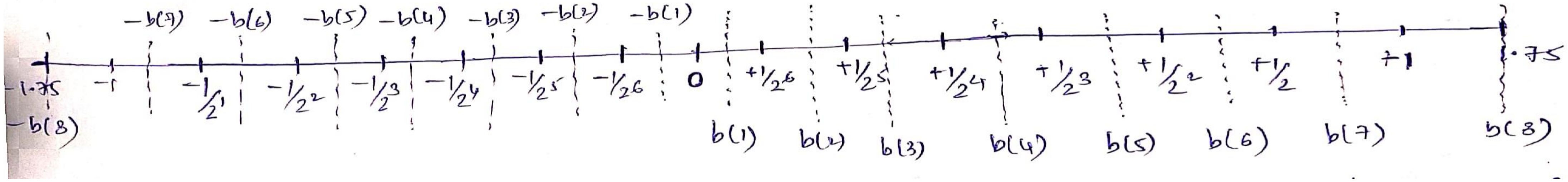
variable	minimum	maximum
input	-5.6046	9.357
weights1	-0.7301	1.0194
weights2	-1.2593	1.2614
weights3	-1.9056	0.9512
h1_ip	-11.3456	10.456
h1_op	0	10.456
h2_ip	-24.2823	22.2721
h2_op	0	22.2721
l_ip	-71.8108	68.1242
l_op	0	1

Total number of bits	bits for decimal part	test accuracy
8	0	24.27
9	1	82.5
10	2	92.9
11	3	95.71
12	4	96.14
13	5	96.32
15	7	96.29
18	10	96.28
20	12	96.31

The best fit would be using 12 bits for the representation with 8 bits for integer and 4 bits for fractional part. For implementation on cortex we can make use of short.

Multiplier Less implementation

- All the weights are in the range of -2 to 2.
- They are further quantised into powers of two so that a multiplication operation can be converted to a shift operation.
- This saves lot of power and time while inference is running.
- Helpful for Edge AI applications.
- Further the weights can now be represented using indexes only(15 different indexes for 15 different weights.) therefore instead of short char datatype can be used.



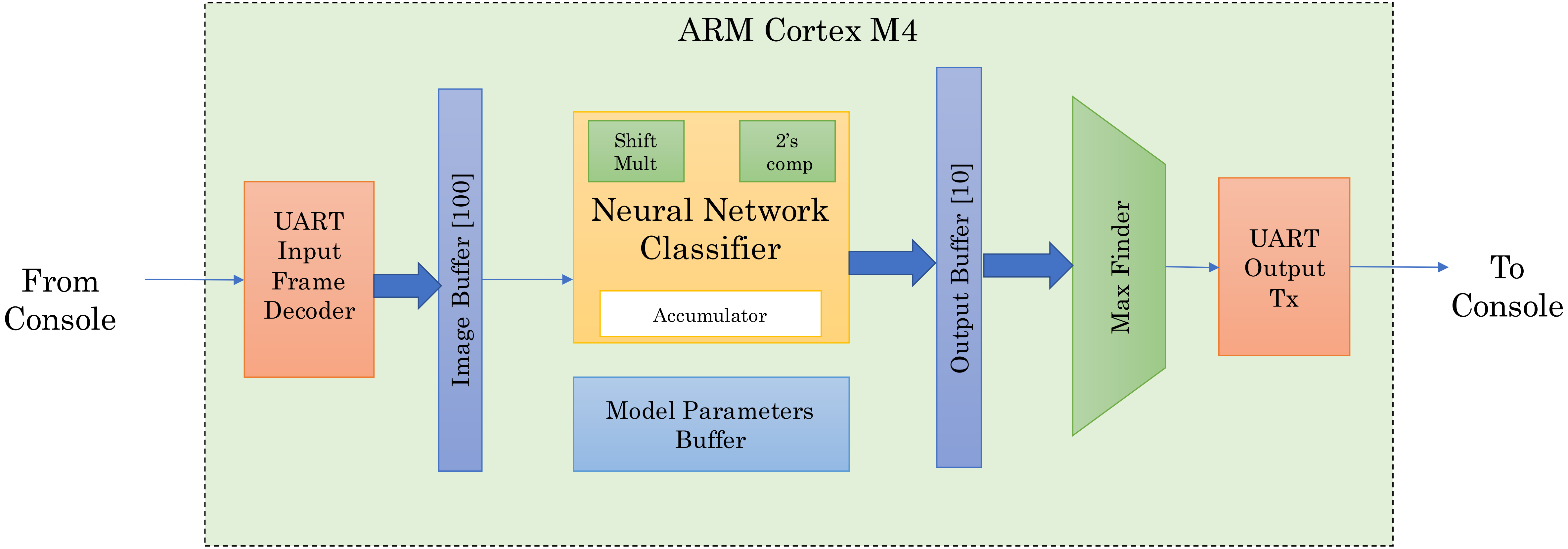
boundary	value
b(1)	0.0078125
b(2)	0.234375
b(3)	0.0390625
b(4)	0.0859375
b(5)	0.1640625
b(6)	0.3359375
b(7)	0.6640625
b(8)	1.75

range of weight	Quantised to
-b(1) to b(1)	0
b(1) to b(2)	$\frac{1}{2^6}$
-b(2) to -b(1)	$\frac{-1}{2^6}$
b(2) to b(3)	$\frac{1}{2^5}$
-b(3) to -b(2)	$\frac{-1}{2^5}$
b(3) to b(4)	$\frac{1}{2^4}$
-b(4) to -b(3)	$\frac{-1}{2^4}$
b(4) to b(5)	$\frac{1}{2^3}$
-b(5) to -b(4)	$\frac{-1}{2^3}$
b(5) to b(6)	$\frac{1}{2^2}$
-b(6) to -b(5)	$\frac{-1}{2^2}$
b(6) to b(7)	$\frac{1}{2^1}$
-b(7) to -b(6)	$\frac{-1}{2^1}$
b(7) to b(8)	1
-b(8) to -b(7)	-1

Board Implementation

- Challenges
  - The microcontroller has very limited memory – 8kB, which is further divided into .data , .bss and stack regions. The original model has ~2,00,000 parameters, which requires about 2,00,000*4 Bytes = 800 kB of SRAM storage all alone.→ Requires Model Compression
  - Neural Network computations are mostly Vectored DSP operations, which are well supported by High end Desktop CPUs and GPUs. Cortex-M microcontrollers do not feature dedicated hardware to accelerate such computations and thus, end up taking long computational times often leading to possible stack overflows. → Techniques such as quantization, multiplier-less optimization, PCA etc.,

Implementation Block Diagram



Data flow

- From a set of 100 *test-image indices,* the user selects one index
- The image corresponding to the user's choice will be displayed on the console.
- The image is packetized by the API and sent over UART.
- The µC decodes the frame header and stores the image in a buffer for further processing. On receipt of entire frame, a flag is set to enable the neural network engine
- The *Neural Network Classifier* outputs the class likelihood values for all classes
- *Max-finder* sends the class having the highest likelihood value over UART back to the host.
- The host displays the identified image on the screen.

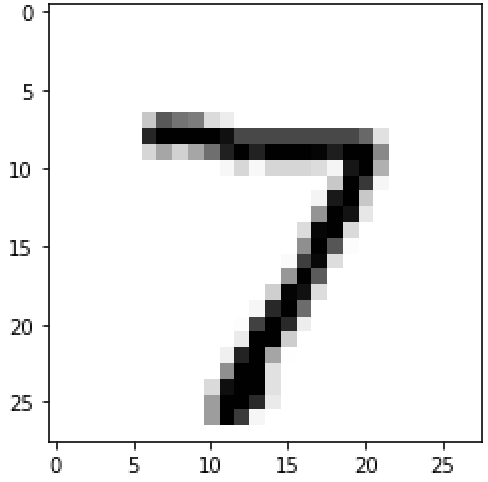
Data Handling

- Each image is of size *100 Bytes*
- Image packetization involves inserting headers to the image {*Header: 0xAA55BB66*}
- Thus, each UART packet is of size 104 Bytes. This information is used by the microcontroller to ensure that it performs classification on valid data only and rejects spurious UART transactions.

API Software

PIC of User Interface

- Offers user interface to enable user to
  - Select an image
  - Display the selected image graphic
  - Displays the identified image for user to compare
- Built in *C* and *Python*
- On receiving image-index from the user, it transfers the image over UART and waits for a maximum of 1 second to receive the classifier output.



Serial Port Configuration

- Uses <termios.h> library
- UART Frame : (`1 start + 8 data + 1 stop + 0 parity`) bits
- Flow Control (Hardware and software) Disabled
- Canonical mode Disabled
- SIGINT chars disabled
- `VMIN = 0, VTIME = 10` → During read, the UART has a time-out of 1 s(10 ds). If no data is received within 1 s, then UART read terminates.
- Baud Rate : 9600 bps

```
*********************WELCOME**************************
*HANDWRITTIN DIGIT RECOGNITION USING ARM CORTEX M4*
*E3-257 Embedded Systems Course Project (2020)     *
*Team : Naveen Chander, Sai Vamshi                 *
***************************************************
*                  INSTRUCTIONS                    *
* Select any Image out of 100 images               *
* -------------------------------------------------*
*                                                  *
*                                                  *
*                                                  *
*                                                  *
*                                                  *
*                                                  *
* Enter a number between 0 to 99                   *
0
The Image Number entered is :0
Read 1 bytes. Image Identified as: 7
*                                                  *
*                                                  *
*                                                  *
*                                                  *
*                                                  *
* -------------------------------------------------*
*                                                  *
*                                                  *
*                                                  *
*                                                  *
*                                                  *
*                                                  *
* Enter a number between 0 to 99                   *
```

Embedded Software Design

- UART0_ISR() :
  - Triggered on receipt of every byte of data over UART0.
  - Contains State Machine which issues a *start* flag on decoding the sequence *0xAA* →*0x55* →*0xBB* → *0x66*.
  - To avoid redundant processing, the state machine disables itself after decoding the header sequence till the next 100 Bytes of data is received.
  - A receive counter is maintained to count up to *100* after the assertion of *start* flag. On reaching a count of 100, a flag to initiate the neural network classifier is set and the classifier is invoked.

Embedded Software Design contd…

- Main()
  - Waits for the *classifier_init* flag from the UART0_ISR
  - Initiates the classification process
  - When classification is complete, it deploys a *max finder* to identify the *most likely* image and sends it over UART0 to the user.

Embedded Software Design contd…

- ann(my_num*,my_num*)
  - This function is called when the *classifier_init* is set by UART0_ISR
  - It takes the input from imagebuf and starts classification using the parameters learnt during training.
  - It is a 4 layer neural network including the input and output layers.
  - It is generally a matrix multiplication followed by non-linear operation at each layer (ReLu in our case.)
  - The number of nodes in each layer are as follows [100 30 16 10].
  - The matrix multiplication in this case is converted to a shift operation because a multiplication operation takes time(as we are targeting for edge AI application.)

Embedded Software Design contd…

- my_mult(my_num*,weight_index,my_num*)

  - It takes the number to be shifted and index as input laong with a pointer to store the result.
  - The number is shifted according to the index and the result passed to calling function using the third argument which is a pointer.