POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Hardware Aware Acceleration For Deep Neural Network

MohammadHossein AskariHemmat

March 6th 2020

POLYTECHNIQUE MONTRÉAL

Presentation outline:

1. Computation Cost in Deep Neural Networks (DNNs)
2. Acceleration In Deep Neural Networks
3. U-NET Fixed-Point Quantization
4. Multi Precision Hardware Accelerator

Computation Cost in Deep Neural Networks (DNNs)

Training Computation Cost :

Finishing a 90-epoch ImageNet-1k training with ResNet-50 on a NVIDIA M40 GPU takes 14 days. This training requires 10^{18} single precision operations in total [Y. You et al "ImageNet Training in Minutes"].

Inference Computation Cost :

Finishing a full pass of Imagenet with input size of 224x224 with batch size of 128 requires 13 GB feature memory and 497 GFLOPs [S. Albanie GitHub:convnet-burden].

Generate completely new images similar to the training images

Figure: Image taken from Karras et al. ICLR2018

Generate completely new images similar to the training images

Figure: Image taken from Karras et al. ICLR2018

Quantization For Accelerating Computation in DNNs :

What is Quantization in DNN?

Quantization is a technique to reduce memory consumption and the computation time of deep neural networks by lowering the precision of parameters.

Benefits;

- Lower power consumption compared to full precision (floating point).
- Faster computation.

Quantization For Accelerating Computation in DNNs :

- Quantization not only uses less memory but it is more energy efficient:

Operation	MUL	ADD
8-bit Integer	0.2pJ	0.03pJ
32-bit Integer	3.1pJ	0.1pJ
16-bit Floating Point	1.1pJ	0.4pJ
32-bit Floating Point	3.7pJ	0.9pJ

Figure: Energy consumption of multiplication and accumulation in a 45nm process (Horowitz, 2014)

- It also speeds up the computation specially for multiplication and division. For instance for Intel Core i7 4770 3.40GHz doing 32-bit multipication is more than 3 times faster for fixed point data types compared to floating point data types [https://goo.gl/7Y7GWt].

U-NET Fixed-Point Quantization Project Collaborators:

MohammadHossein AskariHemmat,
Yvon Savaria, Jean-Pierre David

Sina Honari

Lucas Rouhier, Christian S. Perone,
Julien Cohen-Adad

Published at MICCAI 2019

U-NET For Medical Image Segmentation:

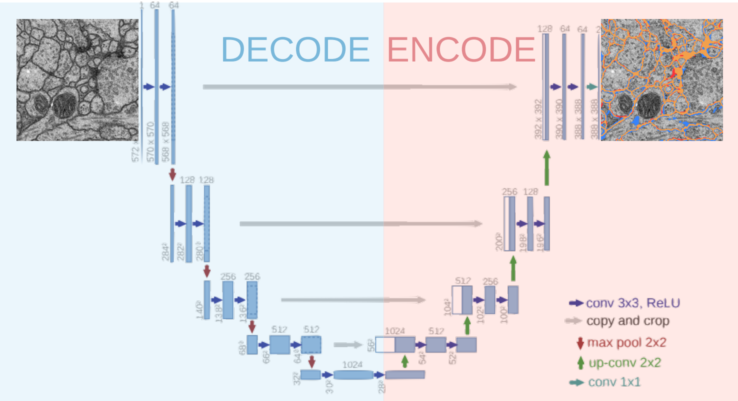


Figure: Taken from O. Ronneberger, 2015

Results For Quantization of U-Net Model for Medical Image:

- We used three different datasets:

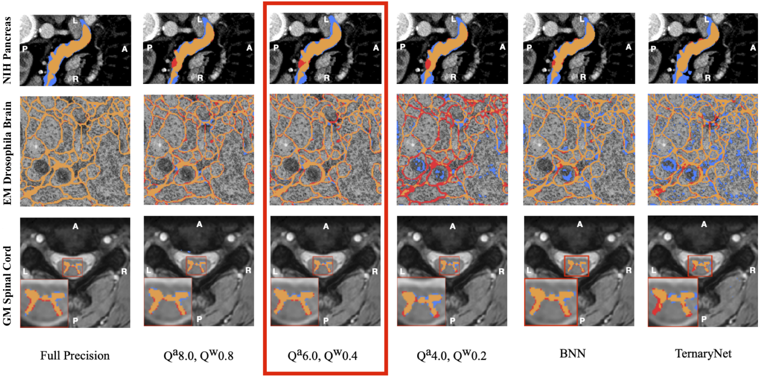


Figure: Segments in blue show false positive, segments in red show false negative and segments in orange show true positive.

Results For Quantization of U-Net Model for Medical Image:

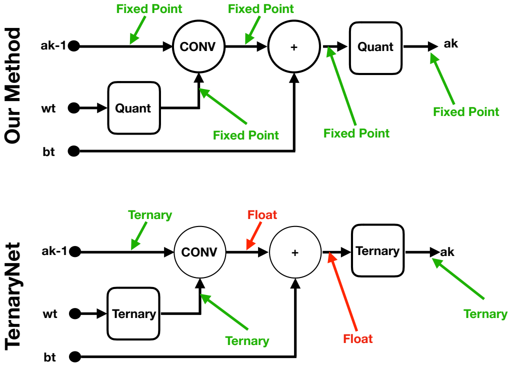
- $Q^a 6.0, Q^w 0.4$ compared to other methods:

Quantization			EM Dataset		GM Dataset		NIH Panceas
Activation	Weight	Parameter Size	Dice Score ReLU	Dice Score Tanh	Dice Score ReLU	Dice Score Tanh	Dice Score
Full Precision		18.48 MBytes	94.05	93.02	56.32	56.26	75.69
Q8.8	Q8.8	9.23 MBytes	92.02	91.08	56.11	56.01	74.61
Q8.0	Q0.8	4.61 MBytes	92.21	88.42	56.10	53.78	73.05
Q6.0	Q0.4	2.31 MBytes	91.03	90.93	55.85	52.34	73.48
Q4.0	Q0.2	1.15 MBytes	79.80	54.23	51.80	48.23	71.77
BNN [18]		0.56 MBytes	78.53	-	31.44	-	72.56
TernaryNet [20]		1.15 MBytes	-	82.66	-	43.02	73.9

Figure: shows best score overall and shows best score between three quantiation methods.

Overall Computation Data Path:

- Fully fixed point data path:



How to run custom precision?

Multi Precision Accelerator Project Collaborators:

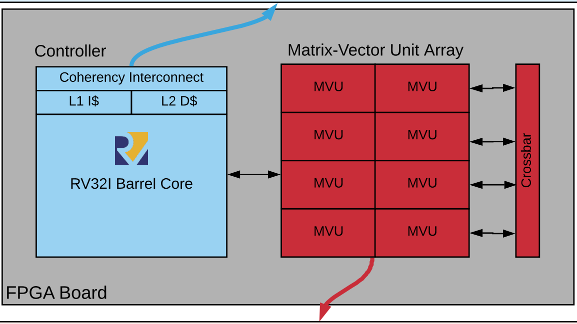
POLYTECHNIQUE MONTRÉAL — MohammadHossein AskariHemmat, Yvon Savaria, Jean-Pierre David

Mila — Olexa Bilaniuk

IBM — Sean Wagner

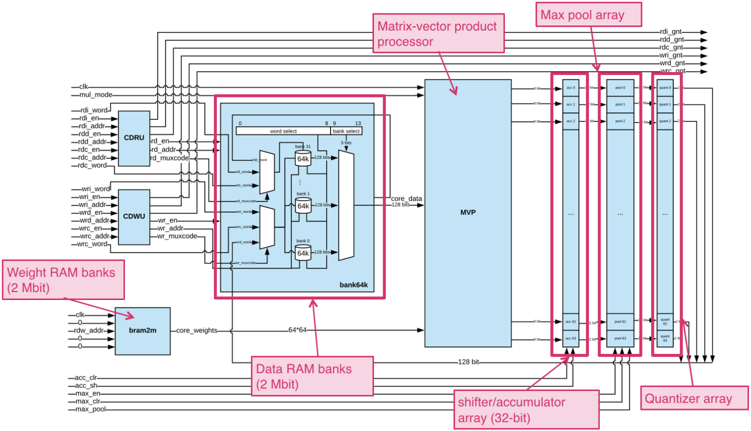
Accelerator Architecture:

RISC-V Barrel Processor For Deep Neural Network Acceleration (FCCM 2020), AskariHemmat et. al

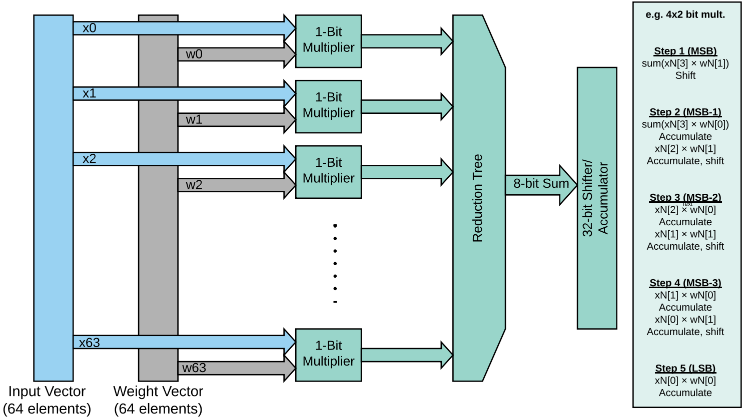


Bit-Slicing FPGA Accelerator for Quantized Neural Networks (ISCAS 2019), Olexa Bilaniuk et. al

Matrix-Vector Unit (MVU):



Bit-serial Vector-Vector Product (VVP):

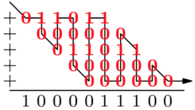


Arbitrary precision with bit-serial math:

```
A=011011,  B=10100;   C=A×B

Step 1 (t=0)        Step 2 (t=1)        Step 3 (t=3)
A:      011011      A:      011011      A:      011011
B:       10100      B:       10100      B:       10100
+            0      +            1      +            1
C:  0000000000      C:  0000000001      C:  0000000011

Step 4 (t=6)        Step 5 (t=10)       Step 6 (t=15)
A:      011011      A:      011011      A:      011011
B:       10100      B:       10100      B:       10100
+            1      +           10      +            1
C:  0000000111      C:  0000010000      C:  0000100001

Step 7 (t=20)       Step 8 (t=24)       Step 9 (t=27)       Step 10 (t=29)
A:      011011      A:      011011      A:      011011      A:      011011
B:       10100      B:       10100      B:       10100      B:       10100
+            1      +            1      +            0      +            0
C:  0001000011      C:  0010000111      C:  0100001110      C:  1000011100
```

Start cycle time

Partial sums



Total time: 30 cycles

Implementation Result on Stratix V GX A7 FPGA:

On it's default configuration (8 MVUs with ternary-binary matrix-vector precision):

- Running at 250MHz
- Consumes 20.883W
- May carry out 250K ternary-binary matrix-vector operations per second, or 8.2 TMAC/s.

Unit	#	ALM
Interconnect	1	3096
MVU	8	19000
Dot-Product	64	12000
Bank Conflict Resolvers	32	5400
Accumulator	64	1300
Max-Pooling	64	2250

Controller:

We designed a Barrel RISC-V Processor:

- Compatible with RV32I Spec.
- We used GNU tool chain ported to RISC-V to program and debug.
- Has 8 Hardware Threads (Harts).
- Has a 5 stage pipe line .
- Runs at 350 MHz with CPI of 1 and consumes 0.287W.

Barrel Processor:

- A barrel processor is a fine-grain multithreading.
- At each clock cycle, we switch between different thread.
- Goal: Maximizing the overall utilization of the processor's resources.

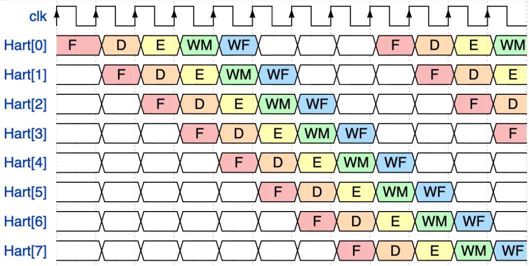
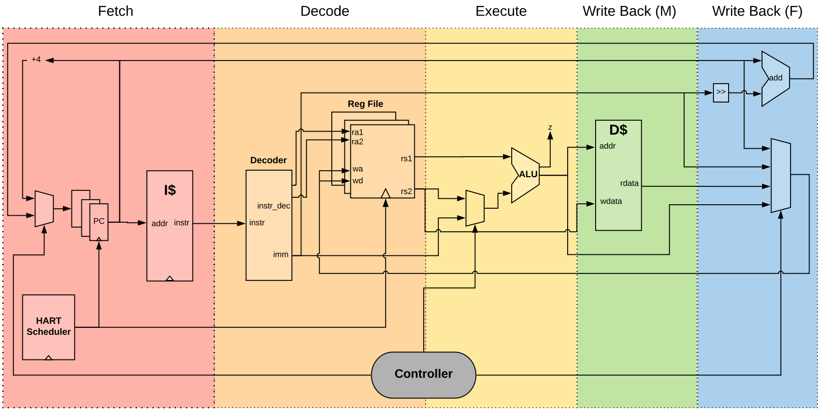


Figure: Thread execution order in a Barrel Processor.

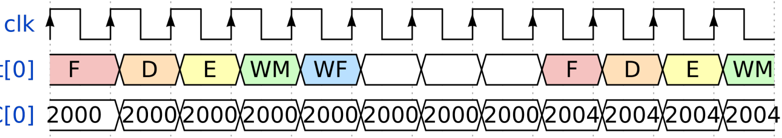
POLYTECHNIQUE MONTRÉAL

Barrel RV32I Core:



Barrel RV32I Core Running Code:

```
00002000 <_start>:
    2000: 00002537                  lui a0,0x2
    2004: 02050513                  addi   a0,a0,32
    2008: 10000637                  lui a2,0x10000

0000200c <.prname_next>:
    200c: 00050583                  lb   a1,0(a0)
    2010: 00058a63                  beqz   a1,2024
    2014: 00b62023                  sw   a1,0(a2)
    2018: 00150513                  addi   a0,a0,1
    201c: ff1ff06f                  j 200c
```



Barrel RV32I Core Implementation Details:

	Baseline	Proposed Architecture	GRVI Phalanx [19]	PicoRV32 Regular [20]
LUT	1111(0.623%)	1698 (0.87%)	320(0.13%)	909(0.51%)
LUTRAM	48(0.04%)	423 (0.38%)	N/A	48(0.04%)
FF	497 (0.1%)	691(0.14%)	N/A	574 (0.12%)
BRAM	15 (2.5%)	15 (2.5%)	N/A	15 (2.5%)
Dynamic Power	0.201W	0.287W	0.043W	0.172W
Frequency	350MHz	350MHz	375MHz	400MHz
CPI	8	1	1.3	4.1

Figure: Implementation on Kintex Ultrascale 40 (KU040).

Conclusion:

- Efficient Deep Neural Networks computation is still a challenge.
- Quantization is a method to accelerate computation but without proper Hardware is not as efficient.
- Quantization can be applied on critical applications (such as medical segmentation) to accelerate computation.
- A custom hardware is needed to do custom precision computation.

POLYTECHNIQUE MONTRÉAL

Thank you for your attention!