“Neural Architecture Search for Tiny Devices”

Philip Leong – Chief Technology Officer, CruxML Pty Ltd
Professor, Computer Systems
School of Electrical and Information Engineering, University of Sydney

April 20, 2023
Thank you, tinyML Strategic Partners, for committing to take tinyML to the next Level, together
Executive Strategic Partners
The Leading Development Platform for Edge ML

dgeimpulse.com
Advancing AI research to make efficient AI ubiquitous

Power efficiency
- Model design, compression, quantization, algorithms, efficient hardware, software tool

Personalization
- Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning
- Robust learning through minimal data, unsupervised learning, on-device learning

Perception
- Object detection, speech recognition, contextual fusion

Reasoning
- Scene understanding, language understanding, behavior prediction

Action
- Reinforcement learning for decision making

A platform to scale AI across the industry
Accelerate Your Edge Compute

SYNTIANT
Making Edge AI A Reality

www.syntiant.com
Platinum Strategic Partners
Renesas is enabling the next generation of AI-powered solutions that will revolutionize every industry sector.
DEPLOY VISION AI
AT THE EDGE AT SCALE
Gold Strategic Partners
Witness potential made possible at analog.com.
Easily deploy your tinyML solutions with Arduino Pro

arduino.cc/pro

Made In Italy
Arm AI Virtual Tech Talks

The latest in AI trends, technologies & best practices from Arm and our Ecosystem Partners.

Demos, code examples, workshops, panel sessions and much more!

Fortnightly Tuesday @ 4pm GMT/8am PT

Find out more: www.arm.com/techtalks
NEUROMORPHIC INTELLIGENCE FOR THE SENSOR-EDGE

www.innatera.com
STMicroelectronics provides extensive solutions to make tiny Machine Learning easy
We engineer exceptional experiences for consumers in the home, at work, in the car, or on the go.

www.synaptics.com
Silver Strategic Partners
Join Growing tinyML Communities:

tinyML - Enabling ultra-low Power ML at the Edge

The tinyML Community
https://www.linkedin.com/groups/13694488/

14k members in
47 Groups in 39 Countries

4k members
&
11.6k followers
Subscribe to tinyML YouTube Channel for updates and notifications (including this video)
www.youtube.com/tinyML
tinyML EMEA Innovation Forum

June 26 - 28, 2023
Amsterdam

EMEA 2023
https://www.tinyml.org/event/emea-2023

More sponsorships are available: sponsorships@tinyML.org
Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums  youtube.com/tinyml

Please use the Q&A window for your questions
Philip Leong

Philip Leong received the B.Sc., B.E. and Ph.D. degrees from the University of Sydney. In 1993 he was a consultant to ST Microelectronics in Milan, Italy working on advanced flash memory-based integrated circuit design. From 1997-2009 he was with the Chinese University of Hong Kong. He is currently Professor of Computer Systems in the School of Electrical and Information Engineering at the University of Sydney, Visiting Professor at Imperial College, and Chief Technology Officer at CruxML Pty Ltd.
Low Precision Inference and Training for Deep Neural Networks

Philip Leong
Director, Computer Engineering Laboratory
http://phwl.org/talks
Focuses on how to use parallelism to solve demanding problems
- Novel architectures, applications and design techniques using FPGAs

Research: reconfigurable computing, radio frequency machine learning
Tradeoff between performance and precision

› CPUs/GPUs designed to support datatypes of fixed wordlength
  - Double, float, long, short, char
› FPGA and ASICs can provide custom datapaths of arbitrary wordlength

<table>
<thead>
<tr>
<th>Precision</th>
<th>Peak TOPS</th>
<th>On-chip weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b</td>
<td>~66</td>
<td>~70 M</td>
</tr>
<tr>
<td>8b</td>
<td>~4</td>
<td>~10 M</td>
</tr>
<tr>
<td>16b</td>
<td>~1</td>
<td>~5 M</td>
</tr>
<tr>
<td>32b</td>
<td>~0.3</td>
<td>~2 M</td>
</tr>
</tbody>
</table>

So how can we utilize low-precision for inference and training?

Slide: Xilinx
› Block Minifloat
› Time series Prediction
› Transfer Learning
Block Minifloat

Sean Fox
Training has greater efficiency problem than inference!
  - E.g. 3x more MACs, much higher memory requirements

Specialized number representations have been proposed
  - Alternatives to FP32/FP16
  - 4-8 bits for weights, activations and gradients
  - Cheaper and faster training systems
  - Focus on Edge (not sure about the Data Center)
Minifloat

- Narrow floating-point representation
  - Our range between 4-8 bits
  - NaN/Infinity NOT supported

Pros:
- Memory (fewer bits)
- Smaller hardware

Cons:
- Dynamic Range (exponent bits)
- Share exponent bias across **blocks** of NxN minifloat numbers

- Dynamic range (with fewer bits)
- Denser dot-products in hardware
- Share exponent bias across **blocks** of NxN minifloat numbers

- Dynamic range (with fewer bits)
- Denser dot-products in hardware

- **Align** with max exponent
- **Underflow** is tolerated
Block Minifloat

Fixed

Minifloat

BFP

Block Minifloat
▪ **Kulisch Accumulator**: Fixed point accumulator wide enough to compute error-free sum of floating-point products

▪ Integer-like hardware complexity for exponent $\leq 4$ bits
Implementation Details

- Three techniques to reduce data loss:
  - Gradual underflow, Block Design, Hybrid Formats

- Simulate specialized BM hardware on GPU (with FP32)
  - Apply Block Minifloat to all weights, acts, grads

- Our Spectrum of Block Minifloats

<table>
<thead>
<tr>
<th>BM8 (ours)</th>
<th>(2,5)/(4,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM7 (ours)</td>
<td>(2,4)/(4,2)</td>
</tr>
<tr>
<td>BM6 (ours)</td>
<td>(2,3)/(3,2)</td>
</tr>
<tr>
<td>BM5 (ours)</td>
<td>(2,2)/(3,1)</td>
</tr>
<tr>
<td>BM5-log (ours)</td>
<td>(4,0)/(4,0)</td>
</tr>
<tr>
<td>BM4 (ours)</td>
<td>(2,1)/(3,0)</td>
</tr>
<tr>
<td>BM4-log (ours)</td>
<td>(3,0)/(3,0)</td>
</tr>
</tbody>
</table>
Data Loss Experiments

(a) Validation Accuracy: Training with denormal numbers on ImageNet

(b) HW (left axis) vs Range (right axis): Selecting the block size

(c) Minifloat scaling by varying the exponent base
End-to-end GPU Training with BM

- Weight, activation and gradient tensors quantized to BM with stochastic rounding
- Kulisch accumulator ensures our dot products are exact (can use FP CUDA lib directly)
- FP32 used for Kulisch to floating-point conversion, block minifloat alignments, quantization etc.
- Approx 1x floating point operation every N MACs, 5x slowdown
Training Experiments (1)

ResNet18 on ImageNet Validation

<table>
<thead>
<tr>
<th>Scheme</th>
<th>BFP (ours)</th>
<th>BM (ours)</th>
<th>▽</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-bit</td>
<td>67.0</td>
<td>69.0</td>
<td>+2.0</td>
</tr>
<tr>
<td>8-bit</td>
<td>69.2</td>
<td>69.8</td>
<td>+0.6</td>
</tr>
</tbody>
</table>
Training Experiments (2)

Transformer on IWSLT’14 DE-En dataset
### Training Experiments Summary

<table>
<thead>
<tr>
<th>Model (Dataset) [Metric]</th>
<th>FP32</th>
<th>BM8</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet (ImageNet)</td>
<td>56.0</td>
<td>56.2</td>
</tr>
<tr>
<td>EfficientNet-b0 (small ImageNet)</td>
<td>62.6</td>
<td>61.8</td>
</tr>
<tr>
<td>LSTM (PTB)[Val ppl.]</td>
<td>84.7</td>
<td>87.33</td>
</tr>
<tr>
<td>Transformer-base (IWSLT)[BLEU]</td>
<td>32.3</td>
<td>31.8</td>
</tr>
<tr>
<td>SSD-Lite (MbNetV2) (VOC)[mAP]</td>
<td>68.6</td>
<td>68.0</td>
</tr>
</tbody>
</table>

Training Accuracy with BM ≈ FP32
RTL Synthesis Results

- Designs synthesized at 750MHz with Cadence RTL Compiler and 28nm cell library
  - Fused multiply-add (FMA)
  - 4x4 systolic matrix multipliers

<table>
<thead>
<tr>
<th>Component</th>
<th>Area ($\mu m^2$)</th>
<th>Power ($\mu W$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP32</td>
<td>4782</td>
<td>10051</td>
</tr>
<tr>
<td>FP8 (w/ FP16 add)</td>
<td>829</td>
<td>1429</td>
</tr>
<tr>
<td>INT8 (w/ INT32 add)</td>
<td>417</td>
<td>1269</td>
</tr>
<tr>
<td>BM8</td>
<td>391</td>
<td>1141</td>
</tr>
<tr>
<td>BM6</td>
<td><strong>200</strong></td>
<td><strong>624</strong></td>
</tr>
<tr>
<td>INT8 (4x4 systolic)</td>
<td>7005</td>
<td>20253</td>
</tr>
<tr>
<td>FP8 (4x4 systolic)</td>
<td>18201</td>
<td>56202</td>
</tr>
<tr>
<td>BM8 (4x4 systolic)</td>
<td><strong>6976</strong></td>
<td><strong>18765</strong></td>
</tr>
</tbody>
</table>

BM8 area and power comparable to INT8
Model: ResNet-18
Dataset: ImageNet

BM units are:
- Smaller
- Consume less Power
Time Series Prediction

Wenjie Zhou
Previous work used GPU implementations with 28nm ASIC study
Here we explore FPGA implementation
- NBEATS Inference and Training implementation using 4-bit mixed-precision BM
- BM GEMM array and Training accelerator architecture for NBEATS
› N-beats: Neural basis expansion analysis for interpretable time series forecasting. ICLR, 2019

› Achieves state of the art time series prediction results

› NN comprises mainly FC layers with shortcut connections
Inference Accelerator Architecture

Vector Addition

Weight

Input

Output

DDR

FPGA

BMVecAdd

IM buffer

BMGEMM+ KulToBM

IFM buffer (A[K])

OFM buffer (C[K])

ExpAlign+ RELU

GEMM
› Each PE performs multiplication and Kulisch accumulation
› Intermediate results are stored in the Kul buffer
› Result transformed to a BM format
M4 competition dataset

Accuracy of BM8 is similar to FP32

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>M4 dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>Yearly, Quarterly, Monthly, Daily</td>
</tr>
<tr>
<td>Training Loss</td>
<td>mean absolute percentage error (MAPE)</td>
</tr>
<tr>
<td>Validation Loss</td>
<td>symmetric mean absolute percentage error (sMAPE)</td>
</tr>
<tr>
<td>Batch size</td>
<td>1024</td>
</tr>
</tbody>
</table>

\[
MAPE = \frac{1}{H} \sum_{i=1}^{H} \frac{|l_i - p_i|}{|l_i|}
\]

\[
SMAPE = \frac{200}{H} \sum_{i=1}^{H} \frac{|l_i - p_i|}{|l_i| + |p_i|}
\]

where \( l_i \) is the label in time step \( i \), and \( p_i \) is the prediction in time step \( i \).
Area of BM8 is similar to INT8 but smaller than FP16
BM8 performance and power is close to INT8
NBEATS Training Accelerator Architecture
Mixed-precision Block Minifloat Training

BM MAC unit (PE)

BM GEMM array
Dataset: M4-Yearly, validation loss: SMAPE loss, block size: 64

<table>
<thead>
<tr>
<th>Loss</th>
<th>Configuration</th>
<th>weight</th>
<th>activation</th>
<th>error</th>
<th>gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM4(1)</td>
<td>14.471649</td>
<td>BM&lt;2,1&gt;</td>
<td>unsigned BM&lt;0,4&gt;</td>
<td>BM&lt;0,3&gt;</td>
<td>BM&lt;0,3&gt;</td>
</tr>
<tr>
<td>BM4(2)</td>
<td>14.463654</td>
<td>BM&lt;2,1&gt;</td>
<td>unsigned BM&lt;0,4&gt;</td>
<td>BM&lt;0,3&gt;</td>
<td>FP32</td>
</tr>
<tr>
<td>BFP8</td>
<td>12.914178</td>
<td>BM&lt;0,7&gt;</td>
<td>BM&lt;0,7&gt;</td>
<td>BM&lt;0,7&gt;</td>
<td>BM&lt;0,7&gt;</td>
</tr>
<tr>
<td>BM8</td>
<td>12.939716</td>
<td>BM&lt;2,5&gt;</td>
<td>BM&lt;2,5&gt;</td>
<td>BM&lt;0,7&gt;</td>
<td>BM&lt;0,7&gt;</td>
</tr>
<tr>
<td>FP32</td>
<td>12.924581</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Transfer Learning

Chuliang Guo
Why might we want to do transfer learning at the Edge?

› Private and secure
  - No personal information uploaded to cloud

› Adapt to changing conditions
  - To deal with non-stationary data

› Size, weight, and power (SWaP)
  - Converge to a good solution faster through pretraining
Back-propagation using SGD

- 3X workload of inference

Fig. 1 CNN training workflow: (1) Conv in forward path, (2) transposed Conv in backward path, (3) dilated Conv in gradient generation, and (4) weight update.

Fig. 2 Non-unit stride Conv, transposed Conv, and dilated Conv [1].

Layer-wise CNN blocks
- Unified bm(2,5) representation
- Non-unit stride Conv support
- Simplified mult/add/MAC
- Fused BN&ReLU

Main blocks
- Unified Conv
- Conv & transposed Conv
- Dilated Conv
- Weight kernel partition

Fig. 3 Overall architecture of the generic training accelerator for layer-by-layer processing. BN and ReLU are fused.
Shortcut addition after BN and ReLu functions (enabling fusing)

Unified bm(2,5) for activations, weights, errors, and gradients (simpler HW)

Full precision accuracy with these changes
Channel tiling accelerator

- Updating last several Conv & FC
  - Shortened back-propagation
  - Reduced BRAM for activations
  - Faster convergence

Fig. 8 Transfer learning example from CIFAR-100 to CIFAR-10.
TABLE III
Resource utilisation of and power the ResNet20 accelerator (with the static power of 30W).

<table>
<thead>
<tr>
<th></th>
<th>CLB</th>
<th>LUT</th>
<th>DSP</th>
<th>BRAM</th>
<th>Vivado(W)</th>
<th>PPS(W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full update</td>
<td>28824</td>
<td>166502</td>
<td>686</td>
<td>1171</td>
<td>8.714</td>
<td>35</td>
</tr>
<tr>
<td>6 Conv+FC</td>
<td>25589</td>
<td>161129</td>
<td>685</td>
<td>671</td>
<td>7.725</td>
<td>34</td>
</tr>
<tr>
<td>2 Conv+FC</td>
<td>21340</td>
<td>129453</td>
<td>621</td>
<td>571</td>
<td>6.779</td>
<td>34</td>
</tr>
</tbody>
</table>

TABLE IV
Resource utilisation and power of the VGG-like accelerator (with the static power of 30W).

<table>
<thead>
<tr>
<th></th>
<th>CLB</th>
<th>LUT</th>
<th>DSP</th>
<th>BRAM</th>
<th>Vivado(W)</th>
<th>PPS(W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full update</td>
<td>20688</td>
<td>119086</td>
<td>614</td>
<td>505</td>
<td>6.824</td>
<td>34</td>
</tr>
<tr>
<td>3 Conv+FC</td>
<td>20489</td>
<td>119740</td>
<td>613</td>
<td>325</td>
<td>6.499</td>
<td>34</td>
</tr>
</tbody>
</table>
Latency Breakdown


ResNet20 on SVHN: 6 Conv + FC update.

ResNet20 on CIFAR10: 2 Conv + FC update.

VGG-like: Full update.

VGG-like on CIFAR10&SVHN: 3 Conv + FC update.
Conclusion
Low-precision formats have wide applicability for inference and training in Edge applications
- Doesn’t necessitate accuracy reduction

Faster Training is possible using BM
- Fewer bits – important for memory-bound
- Narrow exponents – denser MAC in compute-bound

What are the applications?

Thank you!

Philip Leong (philip.leong@sydney.edu.au)
http://phwl.org/talks
Copyright Notice

This multimedia file is copyright © 2023 by tinyML Foundation. All rights reserved. It may not be duplicated or distributed in any form without prior written approval.

tinyML® is a registered trademark of the tinyML Foundation.

www.tinyml.org
Copyright Notice

This presentation in this publication was presented as a tinyML® Talks webcast. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyml.org