Extreme Edge AI on Open Hardware

ACCML@HIPEAC Bologna

20.01.2020

Luca Benini\textsuperscript{1,2}

\textsuperscript{1}Department of Electrical, Electronic and Information Engineering

\textsuperscript{2}Integrated Systems Laboratory
Cloud → Edge → Extreme Edge AI aka TinyML

#1 Customer Question on Amazon.com (out of 1,000+):
1. I don’t want any of my (private, personal) videos on any servers not in my control. Is this possible?

#2 Customer Question on Amazon.com (out of 1,000+):
2. How long does the battery charge last?

Source: www.amazon.com/ask/questions/axia/B01M0H4877

Extreme edge AI challenge
AI capabilities in the power envelope of an MCU: 100mW peak (1mW avg)

E. Gousev, Qcomm research
AI Workloads from Cloud to Edge (Extreme?)

- High OP/B ratio
- Massive Parallelism
- MAC-dominated
- Low precision OK
- Model redundancy

GOP+
MB+
Energy efficiency is THE Challenge

High performance MCUs

Low-Power MCUs

J Pineda, NXP + Updates

1pJ/OP=1TOPS/W

InceptionV4 @1fps in 10mW

Cool… But, HOW??
- As VDD decreases, operating speed decreases
- However efficiency increases \( \rightarrow \) more work done per Joule
- Until leakage effects start to dominate
- Put more units in parallel to get performance up and keep them busy with a parallel workload

ML is massively parallel and scales well (P/S \( \uparrow \) with NN size)
The workhorse: A simple RISC-V pipeline + ISA Extensions

3-cycle ALU-OP, 4-cyle MEM-OP → IPC loss: LD-use, Branch

ISA is extensible *by construction* (great!)

V1  Baseline RISC-V RV32IMC
    HW loops

V2  Post modified Load/Store
    Mac

V3  SIMD 2/4 + DotProduct + Shuffling
    Bit manipulation unit
    Lightweight fixed point *(EML centric)*

XPULP extensions: 25KG → 40KG (1.6x)
PULP-NN: Xpulp ISA exploitation

8-bit Convolution

RV32IMC

N

addi a0,a0,1
addi t1,t1,1
addi t3,t3,1
addi t4,t4,1
lbu a7,-1(a0)
lbu a6,-1(t4)
lbu a5,-1(t3)
lbu t5,-1(t1)
mul s1,a7,a6
mul a7,a7,a5
add s0,s0,s1
mul a6,a6,t5
add t0,t0,a7
mul a5,a5,t5
add t2,t2,a6
add t6,t6,a5
bne s5,a0,1c000bc

N/4

RV32IMCXpulp

lp.setup
p.lw w1, 4(a0!)
p.lw w2, 4(a1!)
p.lw x1, 4(a2!)
p.lw x2, 4(a3!)
pv.sdotp.b s1, w1, x1
pv.sdotp.b s2, w1, x2
pv.sdotp.b s3, w2, x1
pv.sdotp.b s4, w2, x2
end

8-bit SIMD sdotp

HW Loop
LD/ST with post increment

Pooling & ReLu
- HW loop
- LD/ST with post-increment
- 8-bit SIMD max, avg INSNS

9x less instructions than RV32IMC
PULP-NN: Data Reuse in the Register File

CMSIS-NN based Matrix Multiplication Layout: 2x2  PULP-NN Matrix Multiplication Layout: 4x2

RegisterFile of the RI5CY core: 32 general purpose registers

2x2: 43% utilization
4x2: 69% utilization

More Data Reuse & Higher utilization of the RF

Peak Performance (8 cores)
2x2 : 12.8 MAC/cyc
4x2 : 15.5 MAC/cyc
Multiple RI5CY Cores (1-16)
Low-Latency Shared TCDM

Tightly Coupled Data Memory

Mem Mem Mem Mem Mem
Mem Mem Mem Mem Mem

Logarithmic Interconnect

RISC-V core RISC-V core RISC-V core RISC-V core

CLUSTER
DMA for data transfers from/to L2
Shared Icached with private “loop buffer”
Results: RV32IMCXpulp vs RV32IMC

8-bit Convolution Results

Overall Speedup of 75x

PULP-NN: an open Source library for DNN inference on PULP cores

3pJ/OP in fdx22

(ISA does matter😊)
An additional controller is used for I/O

PULPissimo

- Mem Cont
- L2 Mem
- RISC-V core
- I/O

interconnect

Tightly Coupled Data Memory

- Mem
- Mem
- Mem
- Mem
- Mem

DMA

Logarithmic Interconnect

HWCE

RISC-V core

I$
Hardware Processing Engines (HWPEs)

- **Memory**: on the data plane, memory "sees" HWPEs as a set of SW cores.

- **RISCY**: on the control plane, cores control HWPEs as a memory mapped peripheral (e.g. a DMA).

**HWPE** = "Virtual" Core #N+1

**HWPE** = "Virtual" peripheral (e.g. DMA)

**Virtual** Core #N+2

**Virtual** Core #N+3

**Data plane**: Address generation

**TCDM Interconnect**

**Shared Memory**

**Control plane**: HWPE wrapper

**Register File + Control Logic**

**Peripheral Interconnect**

**Cores**
2. Decouple the streaming domain from the shared memory domain; convert streams in 3D-strided memory accesses.

3. Allow "jobs" to be offloaded to the HWCE by the RISC-V cores.

4. Weights for each convolution filter are stored privately.

5. Fine-grain clock gating to minimize dynamic power.

HWCE Sum-of-Products

Each input channel pixel is read $N_{outFeatures}$ times.

Each weight is read once.

$\textbf{foreach}$ (out_feature $y$):
$\textbf{foreach}$ (in_feature $x$):
$y += \text{conv}(W, x)$

Linebuffer: min-bandwidth sliding window

$W$ : $5 \times 5$ 16-bit filters

$N_{outFeatures}$ times

$N_{inFeatures} - 1$ times

$1 \times 16$-bit pixel

$1 \times 16$-bit pixels

$\text{conv}(W, x)$
Cluster performance and energy efficiency on a 64x64 CNN layer (5x5 conv)

Scaled to ST FD-SOI 28nm @ Vdd=0.6V, f=115MHz

**PERFORMANCE**

- 1 core: 10 [MOp/s]
- 4 cores: 40 [MOp/s]
- 16b weights: 1000 [MOp/s]
- 8b weights: 10000 [MOp/s]
- 4b weights: 100000 [MOp/s]

**ENERGY EFFICIENCY**

- 1 core: 3250 [GOp/s/W]
- 4 cores: 84x [GOp/s/W]
- 16b weights: 5x [GOp/s/W]
- 8b weights: 0.3 [pJ/Op]
- 4b weights: 0.3 [pJ/Op]

Now coming: HWCEv5.0 – improves scalability & flexibility @ 3TOPS/W
Two independent clock and voltage domains, from 0-133MHz/1V up to 0-250MHz/1.2V

<table>
<thead>
<tr>
<th>What</th>
<th>Freq MHz</th>
<th>Exec Time ms</th>
<th>Cycles</th>
<th>Power mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>40nm Dual Issue MCU</td>
<td>216</td>
<td>99.1</td>
<td>21 400 000</td>
<td>60</td>
</tr>
<tr>
<td>GAP8 @1.0V</td>
<td>15.4</td>
<td>99.1</td>
<td>1 500 000</td>
<td>3.7</td>
</tr>
<tr>
<td>GAP8 @1.2V</td>
<td>175</td>
<td>8.7</td>
<td>1 500 000</td>
<td>70</td>
</tr>
<tr>
<td>GAP8 @1.0V w HWCE</td>
<td>4.7</td>
<td>99.1</td>
<td>460 000</td>
<td>0.8</td>
</tr>
</tbody>
</table>

4x More efficiency at less than 10% area cost
New Application Frontiers: DroNET on NanoDrone

Pluggable PCB: PULP-Shield
- ~5g, 30×28mm
- GAP8 SoC
- 8 MB HDRAM
- 16 MB HFlash
- QVGA ULP HiMax camera
- Crazyflie 2.0 nano-drone (27g)

Only onboard computation for autonomous flight + obstacle avoidance
no human operator, no ad-hoc external signals, and no remote base-station!
More Efficiency (2): Extreme Quantization

Low(er) precision: 8→4→2

<table>
<thead>
<tr>
<th>Model</th>
<th>Bit-width</th>
<th>Top-1 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18 ref</td>
<td>32</td>
<td>31.73%</td>
</tr>
<tr>
<td>INQ</td>
<td>5</td>
<td>31.02%</td>
</tr>
<tr>
<td>INQ</td>
<td>4</td>
<td>31.11%</td>
</tr>
<tr>
<td>INQ</td>
<td>3</td>
<td>31.92%</td>
</tr>
<tr>
<td>INQ</td>
<td>2 (ternary)</td>
<td>33.98%</td>
</tr>
</tbody>
</table>

SOA INQ retraining

2.2% loss → 0% with 20% larger net

MULT → MUX

Equivalent for 7x7 SoP

RISC-V ISA Extensions for extreme quantization

RI5CY microarchitectural extensions

- Overheads (28nm FDX PULPissimo impl.):
  - Area: ~11% (vs. Ri5CY)
  - Timing Overhead: negligible (integrated in PULPissimo)
  - 8-bit MatMul power overhead: 1.8% (integrated in PULPissimo)
  - GP-app power overhead: 3.5% (integrated in PULPissimo)
From +/-1 Binarization to XNORs

\[ y(k_{out}) = \text{binarize}_{\pm 1} \left( b_{k_{out}} + \sum_{k_{in}} \left( W(k_{out}, k_{in}) \otimes x(k_{in}) \right) \right) \]

\[ \text{binarize}_{\pm 1}(t) = \text{sign} \left( \frac{t - \mu}{\sigma} + \beta \right) \]

\[ \text{binarize}_{0, 1}(t) = \begin{cases} 
1 \text{ if } t \geq -\kappa/\lambda \doteq \tau, \text{ else 0 } & \text{ (when } \lambda > 0) \\
1 \text{ if } t \leq -\kappa/\lambda \doteq \tau, \text{ else 0 } & \text{ (when } \lambda < 0) 
\end{cases} \]

\[ y(k_{out}) = \text{binarize}_{0, 1} \left( \sum_{k_{in}} \left( W(k_{out}, k_{in}) \otimes x(k_{in}) \right) \right) \]

Thresholding

Multi-bit accumulation

Binary product → XOR

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>out</th>
<th>A</th>
<th>B</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
XNE: XNOR Neural Engine

Main unit: binary dot-product and thresholding
Quentin: a XNE-accelerated microcontroller

Quentin in GlobalFoundries 22FDX

XNE area is ~14000 $\mu$m$^2$ (71 KGE, 72% Riscy+FPU)

Private memory is 448 KB SRAM + 3r2w 8 KB SCM

Shared memory is 56 KB SRAM + 8 KB SCM

With SRAMs, max eff
@ 0.65V 8.7 Top/s/W

With SCMs, max eff
@ 0.5V 46.3 Top/s/W

Accuracy Loss is high even with retraining (10%+) → mixed precision
TWN & TCN are also a very appealing alternative (under design)
Binary-based Quantization (BBQ)

Normal NN layer: \( y(k_{out}) = b(k_{out}) + \sum_{k_{in}} (W(k_{out}, k_{in}) \otimes x(k_{in})) \)

Inspired by ABC-Net:

BBQ NN layer: \( y(k_{out}) \approx b(k_{out}) + \sum_{i=0..Q_W} \sum_{j=0..Q_A} \sum_{k_{in}} \alpha_i \beta_j (W_{bin}(k_{out}, k_{in}) \otimes x_{bin}(k_{in})) \)

\( \alpha_i, \beta_j \): power-of-2

Scale Module

Binary NN BinConv module

One quantized NN can be emulated by superposition of power-of-2 weighted \( Q_A \times Q_W \) binary NN

\( Q_W \): weight quantization level
\( Q_A \): activation quantization level
Reconfigurable Binary Engine

\[ y(k_{out}) \approx b(k_{out}) + \sum_{i=0..Q_W} \sum_{j=0..Q_A} \sum_{k_{in}} \alpha_i \beta_j W_{\text{bin}}(k_{out}, k_{in}) \otimes x_{\text{bin}}(k_{in}) \]
Mixed-Precision Quantized Networks

Apply minimum tensor-wise quantization to fit the memory constraints with very-low accuracy drop

mixed-precision quantization rule-based bit precision selection based on memory constraints

Only -2% wrt most accurate INT8 mobilenet (224_1.0) which does not fit on-chip
+8% wrt most accurate INT8 mobilenet fitting on-chip (192_0.5)
+7.5% wrt most accurate INT4 mobilenet (224_1.0) fitting on chip
Diversion: why not a fully hardwired engine?

Flexibility is essential!
Amdhal’s effect!
Mixed precision
NN “zoo”

However....
What about uW «sleep»?

Small always-on network → triggers alarm and video capture/streaming for cloud-based forensics.

JPEG encoding on cluster:
- 7.1x
- 3.9x
- 20fps@50Mhz

Target 30 fps...
Need uW always-on Intelligence

Always-on IO Accelerator!

Tightly Coupled Data Memory

Mem Mem Mem Mem Mem

Logarithmic Interconnect

HWCE

RISC-V core

RISC-V core

RISC-V core

RISC-V core

I$

I$

I$

I$

PULPissimo

IOA

Mem Cont

L2 Mem

RISC-V core

I/O

Ext. Mem

CLUSTER

PULP
Not Only CNNs: Hyper-Dimensional Computing

Low Dimensional Input Data (e.g. 32-bit int)

Mapping

[0 1 0 1 ........................ 1]
[1 1 1 0 ........................ 1]
[1 1 0 0 ........................ 0]
[0 1 1 1 ........................ 1]

High-dimensional
Holographic
Distributed
Pseudorandom with i.i.d. components

[0 1 0 1 ........................ 1]
[1 1 1 0 ........................ 1]
[1 1 0 0 ........................ 0]
[0 1 1 1 ........................ 1]

HD-Encoding

• Component-wise Majority
• XOR
• Permutation

Search Vector

Similarity Search (e.g. Hamming Distance)

[1 1 0 1 ........................ 1]

Prototype Vectors

[0 1 0 1 ........................ 1]
[1 1 1 0 ........................ 1]
[1 1 0 0 ........................ 0]
[0 1 1 1 ........................ 1]
[1 1 1 1 ........................ 1]
[0 1 0 1 ........................ 1]

Highly parallel, fault-tolerant binary operators, assoc-min-distance search

Merge storage & computation i.e. In-memory computing
More efficiency (3): HD-Based smart Wake-Up Module

Specifications

- Area: 670kGE
- Max. Frequency: 3 MHz
- SCM-Memory: 32 kBit
- Module Power Consumption: ~ 15μW

Taped out in 22fdx
Spiking Convolutional Neural Network

- Activity-driven computation
- Event-like (sparse) feature representation
- No processing in absence of input events
- Lightweight pre-processing required on emerging Event-based sensor data (e.g. DVS cameras)
Input spike stream  
64 channel  
(8x8 matrix)  
100 time intervals

Weighted stream  
Conv. (3x3 kernel)

Membrane potential  
LIF state variable

Output spikes  
When Membrane potential exceed the threshold

Leaky Integrate&Fire CSNN layer
SNE Accelerator Architecture

- Digital Leaky Integrate & FIRE neuron model
- Data re-ordering optimized for Convolutional SNN
- Programmable streamer autonomous data fetching
- Data locality: Weight reuse and Neuron state stored locally
- Data re-ordering optimized for Convolutional SNN
- Digital Leaky Integrate & FIRE neuron model

PULP SoC interconnect

- Combinational crossbar: High data routing flexibility and low-latency communication
- High Modularity: Configurable number of LIF neuron blocks Datapath “slices”
First silicon implementation

**ROSETTA SoC**

#### SoC physical implementation
- TSMC65 nm technology
- Chip area 4100μm x 3000μm
- Gates full SoC 6M

#### Accelerator physical implementation
- Area Accelerator 333μm²
- Gates Accelerator ~260k

#### Rosetta’s SNE configuration
- 4x64 time domain multiplexed LIF neurons per slice
- 8kB per data-path slice
- 2 data-path slices

<table>
<thead>
<tr>
<th>Number of Neurons</th>
<th>Memory (Accelerator)</th>
<th>Target Frequency</th>
<th>Performance (estimation)</th>
<th>Synaptic OP</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>16kB</td>
<td>250MHz</td>
<td>1.2 TOP/s</td>
<td>~250 GSOP/s</td>
</tr>
</tbody>
</table>
Enable the extraction of low-level features in a parallel and efficient way by integrating pixel-wise mixed-signal processing circuits on the sensor die to reduce the imager energy costs.

Ultra-Low Power Imaging (GrainCam)

Imager performing spatial filtering and binarization on the sensor die through mixed-signal sensing!

This process naturally reflects the operation of a binarized pixel-wise convolution and can be seen as embedding the first convolutional layer within the image sensor die.

Combinational “Fully Spatial” BNN

Binarized Neural Network for early detection as combinational logic (with registers)

Mixed-Signal Sensing
Focal Plane Binarization

Combinational Digital Logic

Apply XNOR and binarization over input data to produce any of output pixels
Synthesis Results

Synthesis of both models with hard-wired or reconfigurable weights

GF 22nm SOI with LVT cells (typical corner case 0.65V, 25°C)

Table II: Synthesis and Power Results for Different Configurations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16×16 var.</td>
<td>1.17</td>
<td>5.87</td>
<td>12.82</td>
<td>560</td>
<td>2.40</td>
</tr>
<tr>
<td>16×16 fixed</td>
<td>0.46</td>
<td>2.32</td>
<td>12.40</td>
<td>541</td>
<td>1.68</td>
</tr>
<tr>
<td>32×32 var.</td>
<td>5.80</td>
<td>29.14</td>
<td>17.27</td>
<td>754</td>
<td>11.14</td>
</tr>
<tr>
<td>32×32 fixed</td>
<td>2.61</td>
<td>13.13</td>
<td>21.02</td>
<td>918</td>
<td>11.67</td>
</tr>
</tbody>
</table>

↑ Two-input NAND-gate size equivalent: 1 GE = 0.199 μm²
† Fanout-4 delay: 1 FO4 = 22.89 ps

Massive area reduction when hard-wiring the weights:
- XNOR operations reduce to wires or inverter, which can be also shared among different receptive fields
- popcounts also exploits sharing mechanisms

Advanced Synthesis Tools become central to exploit weights and intermediate results sharing to reduce the area occupation

Conclusion

- Near-sensor processing → Energy efficiency pJ/OP and below
  - Ultra-low power architecture and circuits are needed
  - Memory is THE challenge
- TinyML: Inference can be squeezed into mW envelope
  - Non-von-Neumann acceleration → remove lmem bottleneck
  - Very robust at low precision → memory footprint reduction
  - fJ/OP is in sight! (100+TFLOPs/W) → mW inference engines!
- Pushing on the memory+IO bottleneck
  - TCDM + SCM memory hierarchy optimazion and logic+physical optimizazion
  - In-place, in memory, staged, event-based, non-DNN inference
  - Better memories and memory interfaces (NVM, HBM…)
  - Multi-chip Systolic (2.5D chiplets)
  - Fature map compression

Open-Source Innovation ecosystem!
The fun is just beginning
### Closing the Bin/Ternarization accuracy Gap

#### Table 1: Experimental Results on ImageNet

<table>
<thead>
<tr>
<th>Model</th>
<th>Method*</th>
<th>Levels†</th>
<th>Accuracy [%] (top-1/top-5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>baseline</td>
<td>full-prec.</td>
<td>69.76/89.08</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>QN</td>
<td>(Yang et al., 2019) 5: {α_i}, {±2^i}</td>
<td>69.90/89.30</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>ADMM</td>
<td>(Leng et al., 2018) 4: {±2^i}</td>
<td>67.50/87.90</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>LQ-Nets</td>
<td>(Zhang et al., 2018) 3: { α_1, α_2, α_3 }</td>
<td>68.00/88.00</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>QN</td>
<td>(Yang et al., 2019) 3: { α_1, α_2, α_3 }</td>
<td>69.10/88.90</td>
</tr>
<tr>
<td>ResNet-18+</td>
<td>TTQ</td>
<td>(Zhu et al., 2017) 3: { α_1, 0, α_2 }</td>
<td>66.60/87.20</td>
</tr>
<tr>
<td>ResNet-18+</td>
<td>ADMM</td>
<td>(Leng et al., 2018) 3: {-1, 0, 1}</td>
<td>67.00/88.00</td>
</tr>
<tr>
<td>ResNet-18+</td>
<td>INQ</td>
<td>(Zhou et al., 2017) 3: {-1, 0, 1}</td>
<td>66.00/88.00</td>
</tr>
<tr>
<td>ResNet-18+</td>
<td>TNN</td>
<td>(Li et al., 2016) 3: {-1, 0, 1}</td>
<td>65.30/86.20</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>RPR (ours)</td>
<td>3: {-1, 0, 1}</td>
<td>66.31/87.84</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>ADMM</td>
<td>(Leng et al., 2018) 2: {-1, 1}</td>
<td>64.60/86.20</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>XNOR-net BWN (Rastegari et al., 2016) 2: {-1, 1}</td>
<td>60.80/83.00</td>
<td></td>
</tr>
<tr>
<td>ResNet-18</td>
<td>RPR (ours)</td>
<td>2: {-1, 1}</td>
<td>64.62/86.01</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>baseline</td>
<td>full-prec.</td>
<td>76.15/92.87</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>ADMM</td>
<td>(Leng et al., 2018) 3: {-1, 0, 1}</td>
<td>72.50/90.70</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>TWN</td>
<td>(Li et al., 2016) 3: {-1, 0, 1}</td>
<td>65.60/86.50</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>RPR (ours)</td>
<td>3: {-1, 0, 1}</td>
<td>71.83/90.28</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>ADMM</td>
<td>(Leng et al., 2018) 2: {-1, 1}</td>
<td>68.70/88.60</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>XNOR-net BWN (Rastegari et al., 2016) 2: {-1, 1}</td>
<td>63.90/85.10</td>
<td></td>
</tr>
<tr>
<td>ResNet-50</td>
<td>RPR (ours)</td>
<td>2: {-1, 1}</td>
<td>65.14/86.31</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>baseline</td>
<td>full-prec.</td>
<td>69.78/89.53</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>ADMM</td>
<td>(Leng et al., 2018) 3: {-1, 0, 1}</td>
<td>63.10/85.40</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>TWN</td>
<td>(Li et al., 2016) 3: {-1, 0, 1}</td>
<td>61.20/84.10</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>RPR (ours)</td>
<td>3: {-1, 0, 1}</td>
<td>64.88/86.05</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>ADMM</td>
<td>(Leng et al., 2018) 2: {-1, 1}</td>
<td>60.30/83.20</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>XNOR-net BWN (Rastegari et al., 2016) 2: {-1, 1}</td>
<td>59.00/82.40</td>
<td></td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>RPR (ours)</td>
<td>2: {-1, 1}</td>
<td>62.01/84.83</td>
</tr>
</tbody>
</table>

**Soa results (Sept19)**

1. First and last layer FP
2. ResNets have type-B bypasses (with 1x1 conv. in the non residual paths)
3. Modified network 2.25x more weights