Fast Implementation of 4-bit Convolutional Neural Networks for Mobile Devices

Anton Trusov[1,3], Elena Limonova[1,2,3], Dmitry Slugin[2,3], Dmitry Nikolaev[2,3,4], Vladimir V. Arlazarov[2,3]

[1] Moscow Institute of Physics and Technology
[2] FRC CSC RAS
[3] Smart Engines Service LLC
[4] Institute for Information Transmission Problems RAS

1. Introduction

Low-bit quantized neural networks (QNNs) allow us to:

- accelerate inference;
- decrease model size;
- perform real-time computation on low-powered devices;
- follow paradigm of edge intelligence.

2. Problem

However, low-bit QNNs do not suit end devices of general architecture well:

- CPUs allow only 8-bit (or multiple) access and computations;
- no efficient CPU implementations for lower than 8-bit quantization.

3. Our contribution

- We provide a novel algorithm for fast inference of 4-bit quantized neural network on CPU, based on a fast multiplication (used in convolution and fully-connected layers).
- We experimentally prove its efficiency for ARM architecture.

4. Quantization scheme

Linear quantization method:

$$\hat{w}_i = \left\lfloor \frac{w_i}{s} \right\rfloor - z$$
$$s = \frac{\max(\max_i w_i, 0) - \min(\min_i w_i, 0)}{2^p - 1}$$
$$z = \min(\min_i w_i, 0),$$

where $\hat{w}_i$ denotes quantized values, w_i are floating-point values, s is scale factor, z is a zero-point (offset), p is a number of bits used in quantized values

5. Quantized multiplication

Let's consider the quantized approximation of matrix multiplication $R = WX$:

$$r_{ij} = \sum_{k=1}^{D} w_{ik} x_{kj}$$
$$\approx \sum_{k=1}^{D} s_w(\hat{w}_{ik} - z_w) s_x(\hat{x}_{kj} - z_x)$$
$$= s_w s_x \Big(\sum_{k=1}^{D} \hat{w}_{ik}\hat{x}_{kj} - z_w \sum_{k=1}^{D} \hat{x}_{kj} -$$
$$- z_x \sum_{k=1}^{D} \hat{w}_{ik} + D z_x z_w \Big)$$

where r_{ij} denotes values of R matrix, w_{ik} and x_{kj} are values of W and X matrices, $\hat{w}_{ik}$ and $\hat{w}_{ik}$ are their quantized approximations, s_w and s_x are scale factors, z_w and z_x are zero-points and D is a depth of multiplication.

6. Reordering

RHS

1	3	5	7	...
2	4	6	8	...
9	11	13	15	...
10	12	14	16	...
...	...	...	...	...

Figure 1: The order or values of right temporal buffer.

LHS

1	9	17	25	...
2	10	18	26	...
3	11	19	27	...
4	12	20	28	...
5	13	21	29	...
6	14	22	30	...
7	15	23	31	...
8	16	24	32	...

Figure 2: The order or values of left temporal buffer.

7. Multiplication micro-kernel

lhs are 128-bit SIMD registers with 16 4-bit quantized values (zero padded to 8-bit).
rhs are 64-bit SIMD registers with 8 4-bit quantized values (zero padded to 8-bit).
res are 128-bit SIMD registers with 8 16-bit quantized values.

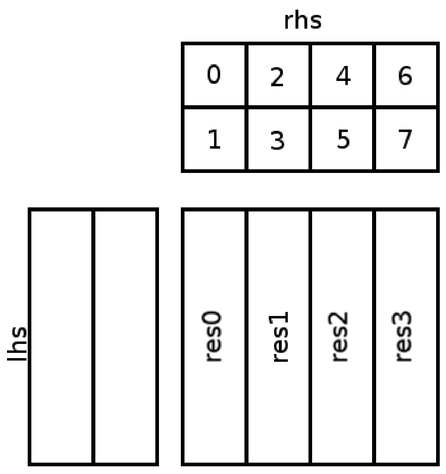

Figure 3: The smaller micro-kernel.

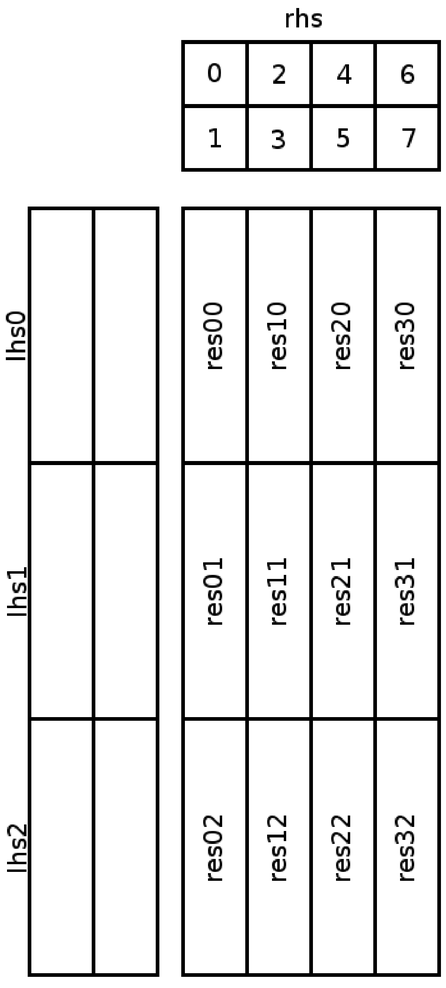

Figure 4: The bigger micro-kernel.

8. Multiplication Algorithm

```
pack right matrix into RHS
pack left matrix into LHS
for j in 0, ..., cols / nr
  {res0 ...  res3} ← next result block
  for i in 0, ..., rows / mr
    for k in 0, ..., depth / 2
      lhs ← next block from LHS
      rhs ← next block from RHS
      VMAL(dst0, LOW(lhs), rhs[0]);
      VMAL(dst0, HIGH(lhs), rhs[1]);
                ...
      VMAL(dst3, LOW(lhs), rhs[6]);
      VMAL(dst3, HIGH(lhs), rhs[7]);
  result block ← {res0 ...  res3}
```

9. Quantized convolutional layer

- Perform Im2col transformation to turn convolution into matrix multiplication.
- Compute matrix multiplication:

$$\hat{r}_{ij} = \sum_{k=1}^{D} \hat{w}_{ik}\hat{x}_{kj} - z_w \sum_{k=1}^{D} \hat{x}_{kj} - z_x \sum_{k=1}^{D} \hat{w}_{ik} +$$
$$+ D z_x z_w,$$

- Save floating-point scale factor: $s_r = s_w s_x$

10. Experiments

- 36 MRZ character recognition from MIDV-500 dataset
- ODROID-XU4 single-board computer with Samsung Exynos5422 ARM processor

Table 1: The network architecture, where F is a number of filters.

Layer	F	Filter size	Stride
Conv + ReLU	8	5×5	1×1
Conv + ReLU	8	3×3	1×1
Conv + ReLU	8	3×3	2×2
Conv + ReLU	16	3×3	1×1
Conv + ReLU	16	3×3	2×2
Conv + ReLU	24	3×3	1×1
FC + SoftMax	36 neurons		

Table 2: Accuracy on synthetic data (AS), accuracy on MIDV-500 (AM), convolution inference time (T_c) and full interence time (T) evaluation.

Model	**AC**, %	**AM**, %	T_c, ms	T, ms
CNN	99.8	95.6	0.99	1.22
QNN-8	99.7	95.4	0.55	0.74
QNN-4	99.2	95.0	**0.45**	**0.63**
QNN-32	-	-	1.16	1.47

11. Results

- Our 4-bit quantized matrix multiplication works about 3 times faster than floating-point multiplication from Eigen library and 1.5 times faster than 8-bit quantized multiplication similar to gemmlowp library
- Our 4-bit QNN works about 2 times faster than traditional CNN and 1.2 times faster than 8-bit QNN of the same architecture.
- The real-world problem of OCR recognition on the MIDV-500 dataset demonstrates 95.0% accuracy, while the floating-point network gives 95.6% accuracy.