Week 4 Report
Qian Jiang

1. Summarized optimization problems and methods for quantization neural networks

    1) Training Quantized Nets: A Deeper Understanding

        Sum: Problem of low-precision weight: stochastic gradient →Small updates of weights → easy to be rounded off by binarization.

        This paper did 1)convergence analysis for classical Stochastic Rounding(SR) & Binary Connect(BC), showing they both are capable of solving convex discrete problems 2) explain why BC works better than SR for non-convex problem

    2) Deep Learning with Limited Numerical Precision

        Sum: Idea is to use algorithm-level noise tolerance to leverage hardware requirements → Co-optimized system

        This paper did 1) 16-bit fixed point number representation using stochastic rounding 2) Proposed a hardware implementation

    3) Training and Inference with Integers in Deep Neural Networks

        Sum: Problem: how to quantize both operands and operations & how many bits are needed for SDG

        This paper proposed WAGE method to discretize both training and inference: 1) For operands, linear mapping & orientation-preserved shifting; For operations, replace batch normalization by constant scaling factor and simplify unnecessary techniques 2) For inference, accumulate-compare cycles; For training, low-bitwidth multiply-accumulate cycles

(Detailed notes attached below)

2. Read code of HWGQ

    Set up environments and learn caffe

Training Quantized Nets: A Deeper Understanding

convex optimization: where all the constraints are convex functions (eg )

Key: training methods for QNN

1. accuracy guarantee for methods under convexity assumptions.
2. explore algorithm behavior for non-convex problem

1. Intro

Problem of low-precision weights:

stochastic gradient method → small updates of w

⇓

easy to be rounded off by binarization/discretization

this paper

① convergence analysis: {classical stochastic rounding, Binary Connect ✱}

↓

Capable of solving convex discrete problems (depends on quantization level)

② why algorithm "maintain floating-point (like BC) work well.

2. fully quantized (like SR) stall before training is complete

⇓

long-term behavior of BC has important property needed for non-convex optimization which SR lacks.

2. Background & Related Work

backprop = back propagation

- quantizing pre-trained models with/without retraining
- training quantized model from scratch
  - ↳ what this paper focus on

3. Training QNN

$$\min F(w) := \frac{1}{m}\sum_{i=1}^{m} f_i(w) \quad (1)$$

$$GSD: \; w^{t+1} = w^t - \alpha_t \nabla \tilde{f}(w^t) \quad (2)$$

$$\hat{w} = Q(w)$$

⇒ ① Deterministic Rounding:

$$Q_d(w) = \text{sign}(w) \cdot \Delta \cdot \left[\frac{|w|}{\Delta} + \frac{1}{2}\right] \quad (3)$$

$$W_b^{t+1} = Q_d\left(W_b^t - \alpha_t \nabla \tilde{f}(W_b^t)\right) \quad (4)$$

② Stochastic Rounding: (random number)

$$Q_s(w) = \Delta \cdot \begin{cases} \lfloor \frac{w}{\Delta} \rfloor + 1, & \text{for } p \le \frac{w}{\Delta} - \lfloor \frac{w}{\Delta} \rfloor \\ \lfloor \frac{w}{\Delta} \rfloor, & \text{otherwise} \end{cases} \quad (5)$$

$$\underset{\downarrow}{E}[Q_s(w)] = w$$

expectation

$$W_b^{t+1} = Q_s\left(W_b^t - \alpha_t \nabla \tilde{f}(W_b^t)\right) \quad (6)$$

③ Binary Connect:

full-precision buffer W_r

$$W_r^{t+1} = W_r^t - \alpha_t \nabla \tilde{f}(Q(W_r^t)) \quad (7)$$

quantizes weights just before the gradient computation

Deep learning with Numerical Precision

Key: ① 16-bit fixed point number representation using stochastic rounding
⇒ little degradation in classification acc
② Hardware.

low-precision : less than 8-bits

1. Intro

idea: algorithm-level noise tolerance
↕ leverage
hardware requirements

⇒ co-optimized system
allowing low-level hardware components to perform approximate, non-deterministic computations

step 1: low-precision fixed-point arithmetic
2: stochastic rounding scheme
3: test on MNIST. CIFAR-10
4: propose a hardware accelerator on FPGA

2. Related Works

① implementing just the feed-forward.
② training using different number representations (most recent 32-bit, this paper 16-bit)

3. Limited Precision Arithmetic

$[Q_I \cdot Q_F]$
Q_I: integer, Q_F: fraction
$IL + FL = WL$

ϵ: smallest positive number can be represented
$= 2^{-FL}$

① Rounding Modes

1. Round to nearest

$$= \begin{cases} \lfloor x \rfloor & , \text{if } \lfloor x \rfloor \le x \le \lfloor x \rfloor + \frac{\epsilon}{2} \\ \lfloor x \rfloor + \epsilon & , \text{if } \lfloor x \rfloor + \frac{\epsilon}{2} < x \le \lfloor x \rfloor + \epsilon \end{cases}$$

2. Stochastic rounding

$$= \begin{cases} \lfloor x \rfloor & , \text{w.p.} \left(1 - \frac{x - \lfloor x \rfloor}{\epsilon}\right) \\ \lfloor x \rfloor + \epsilon & , \text{w.p.} \left(\frac{x - \lfloor x \rfloor}{\epsilon}\right) \end{cases}$$

$$\left(E\left(\text{Round}\left(x, \langle IL, FL \rangle\right)\right) = x \right)$$

$$\text{Convert}(x, \langle IL, FL \rangle) = \begin{cases} -2^{IL-1} & , \text{if } x \le 2^{IL-1} \\ 2^{IL-1} - 2^{-FL} & , \text{if } x \ge 2^{IL-1} - 2^{-FL} \\ \text{Round}(x, \langle IL, FL \rangle) & , \text{otherwise.} \end{cases}$$

② Multiply and accumulate (MACC) operation

$$c_0 = a \cdot b$$

$[\tilde{IL}, \tilde{IF}]$ $[IL, IF]$

step 1: $z = \sum_{i=1}^{d} a_i b_i$ (L of $z \le \log_2 d + 2WL$)

step 2: Convert

$$c_0 = \text{Convert}(z, \langle \tilde{IL}, \tilde{IF} \rangle)$$

Pros: 1. mimic the hardware behavior

2. reduce hardware overhead.

3. efficiently simulate fixed-point computations using CPU/GPUs and vendor-supplied BLAS

4. Training Deep Networks.

constrained parameters: $W^l, B^l, Y^l, \delta^l, \Delta W^l, \Delta B^l$)

unchanged: w initialization, learning rate ...

Training and Inference with Integers in Deep NNs.

Key: 1. WAGE : discretize both training and inference
2. replace batch normalization by constant scaling layer

training ↓ inference

1- Intro

① how to quantize all the operands and operations
② how many bits or states are needed for SGD

WAGE: weight. Activation. Gradient. Error

step 1: linear mapping. orientation-preserved shifting are applied for operands → ternary weights
8-bit integers for A·G

2: batch normalization replaced by constant scaling factor for operations.

3. Simplify other techniques (L2 regularization ...)

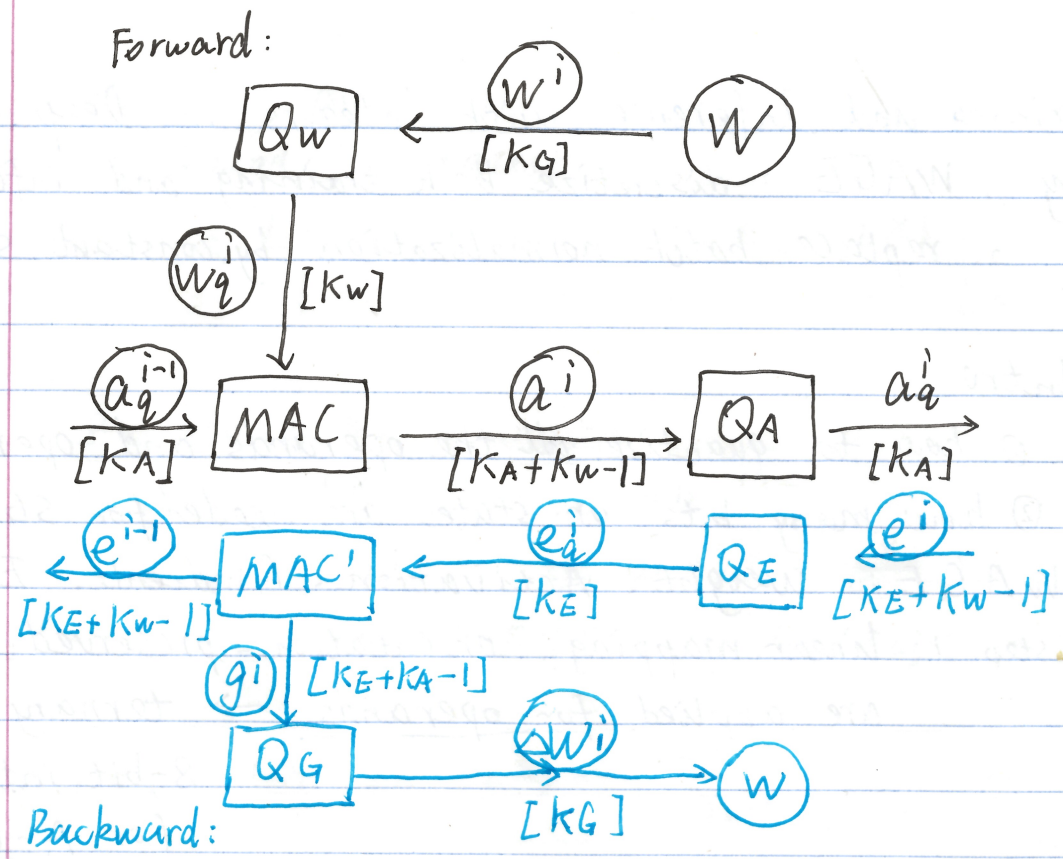
Streamline: inference → accumulate - compare
training → low-bitwidth multiply - accumulate

2. Related Work

① Weight and activation: BC. BNN
② Gradient computation and accumulation: DoReFa-Net

3. WAGE Quantization

$$e^i = \frac{\partial L}{\partial a^i} \cdot g^i = \frac{\partial L}{\partial w^i}$$ (L: Loss function)



operator Q : reduce precision

① Linear-mapping & Stochastic Rounding

$$\sigma(k) = 2^{1-k}, \quad k \in N+$$

$$Q(x,k) = clip\{\sigma(k) \cdot round[\frac{x}{\sigma(k)}], -1+\sigma(k), 1-\sigma(k)\}$$

$$Shift(x) = 2^{round(\log_2 x)}$$

② Weight Initialization

$$W \sim U(-L,+L), L = max\{\sqrt{6/n_{in}}, L_{min}\}$$

$$(L_{min} = \beta\sigma)$$

③ Quantization Details

$$W_q = Q_W(W) = Q(W, k_W) \quad (\alpha = \text{scaling factor} = max\{shift\frac{L_{min}}{L}, 1\}$$

$$a_q = Q_A(a) = Q(\frac{a}{\alpha}, k_A) \quad e_q = Q_E(e) = Q(\frac{e}{shift\{max|e|\}}, k_E)$$

$$g_s = \eta \cdot g / shift(max\{|g|\}) \quad \Delta W \cdot W_{t+1} \cdots$$