ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 18-04-36
230

j.inst.Korean.electr.electron.eng.Vol.22,No.4,1115~1122,December 2018
http://dx.doi.org/10.7471/ikeee.2018.22.4.1115

1-Bit 합성곱 신경망을 위한 정확도 향상 기법
Accuracy Improvement Method for 1-Bit Convolutional Neural Network

임 성 훈*, 이 재 흥*★

Sung-Hoon Im*, Jae-Heung Lee*★

Abstract

In this paper, we analyze the performance degradation of previous 1-Bit convolutional neural network method and introduce ways to mitigate it. Previous work applies 32-Bit operation to first and last layers. But our method applies 32-Bit operation to second layer too. We also show that nonlinear activation function can be removed after binarizing inputs and weights. In order to verify the method proposed in this paper, we experiment the object detection neural network for korean license plate detection. Our method results in 96.1% accuracy, but the existing method results in 74% accuracy.

요 약

본 논문에서는 기존 1-Bit 합성곱 신경망의 성능 하락에 대한 분석과 이를 완화하기 위한 방안을 제시한다. 기존의 연구는 첫 번째 층과 마지막 층만 32-Bit 연산을 적용하고 나머지 연산은 1-Bit 연산을 적용한 것과 달리 본 논문에서는 두 번째 층도 32-Bit로 연산한다. 또한 입력과 가중치를 이진화하고 1-Bit 연산을 적용한 후에는 비선형 활성화 함수를 제거할 수 있음을 제시한다. 본 논문에서 제시한 방법을 검증하기 위해 차량 번호판 검출을 위한 객체 검출 신경망을 실험하였다. 기존의 방법으로 학습한 결과보다 정확도가 74%에서 96.1%로 상승하였다.

Key words : XNOR Neural Network, 1-Bit Neural Network, Quantized Neural Network, Convolutional Neural Network, Neural Network

Ⅰ. 서론

최근 합성곱 신경망의 성능이 다양한 기법에 의해 향상되어 왔다. 대부분의 연구는 신경망의 깊이를 깊게 쌓아 성능을 향상시켰다.

신경망의 영상 분류 능력이 사람 수준으로 근접하게 따라오고 있지만 신경망을 이용한 기술은 매우 많은 연산을 처리해야하기 때문에 고성능 서버에서 서비스되고 있다.

이러한 제약사항이 발생하는 이유는 신경망의 파라미터 수가 많아서 외부 메모리로부터의 읽기/쓰기가 빈번하게 요구되기 때문이다. 외부 메모리는 Dynamic Random Access Memory(DRAM)을 사용하며 Central Processing Unit(CPU) 내부에

* Dept. of Computer Engineering,
★ Corresponding author
E-mail : jhlee@hanbat.ac.kr, Tel : +82-42-821-1208
※ Acknowledgment
This research was supported by The Leading Human Resource Training Program of Regional Neo industry through the National
Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and future Planning (No. 2016H1D5A1911149).
Manuscript received Dec. 7, 2018; revised Dec. 17, 2018; accepted Dec. 18, 2018

있는 캐시 메모리보다 수백 배 느리다. 또한 신경망의 연산은 실수 곱셈과 덧셈으로 이루어져있기 때문에 이러한 연산을 줄이는 것이 스마트폰과 같은 모바일 CPU에서 구동할 수 있도록 하는 핵심이 된다.

앞서 언급한 제약사항을 완화하기 위해 32-Bit 실수 표현 체계를 1-Bit로 변경하는 방법[1]이 존재 한다. 하지만 [1]의 연구는 입력과 가중치를 모두 1-Bit로 변경하였을 때 10% 이상의 성능 저하가 발생한 것으로 보고되었다. 본 연구에서는 이러한 성능 저하를 막기 위한 방법을 제시한다.

이번 Ⅰ장을 시작으로 Ⅱ장에선 신경망의 수 표현 체계와 관련한 연구를 소개하고 Ⅲ장에서는 본 연구에서 제안하는 방법을 소개한다. Ⅳ장 과 Ⅴ장에서는 제안하는 방법을 이용한 실험 결과와 결론에 대해 설명한다.

Ⅱ. 관련연구

1. 8-Bit(정수) 신경망

8-Bit 정수 연산 신경망은 [2]의 방법이 존재한다. 신경망 학습 후에 32-Bit 실수 파라미터를 8-Bit 정수 영역으로 변형하는 것으로 수식 1과 같다. 수식 2와의 차이점은 0의 값을 정확하게 표현하는 것에 있다. 수식 2는 0을 정확하게 표현이 불가능하며 신경망에서는 입력의 크기와 출력의 크기를 같게 맞추기 위해 0의 값을 추가하여 연산하기 때문에 8-Bit 정수로 변환 시에 수식 1을 사용한다.

$$r = S(q - Z) \tag{1}$$

$$r = S \bullet q - Z \tag{2}$$

수식 (1)을 이용하여 행렬 곱을 표현하면 수식 (3)과 같다.

$$S_3(q_3^{(i,k)} - Z_3) = \sum_{j=1}^{N} S_1(q_1^{(i,j)} - Z_1) S_2(q_2^{(j,k)} - Z_2) \tag{3}$$

수식 (3)을 풀어 쓰면 수식 (4)와 같다. 32-Bit 실수 연산과 달리 8-Bit 정수로 변경되었기 때문에 수식 (4)의 $\sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)}$에서 정수 곱셈이 발생한다.

$$q_3^{(i,k)} = Z_3 + M(NZ_1Z_2 - Z_1a_2^{(k)} - Z_2a_1^{(i)} + \sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)}) \tag{4}$$

where

$$M = \frac{S_1S_2}{S_3},\ a_2^{(k)} = \sum_{j=1}^{N} q_2^{(j,k)},\ a_1^{(i)} = \sum_{j=1}^{N} q_1^{(i,j)} \tag{5}$$

8-Bit 정수 신경망은 성능 하락의 폭이 크지 않고 기존 32-Bit에 비해 메모리 접근이 4배 줄어들게 되며 정수 연산으로 변경되기 때문에 신경망 연산 속도가 2~3배 증가하게 된다.

2. 2-Bit(삼진) 신경망

삼진 신경망은 파라미터를 -1, 0, +1로 변경하여 연산한다. 대표적으로 [3]의 방법이 있다. 32-Bit 파라미터를 -1에서 +1 사이로 정규화하고 임계값 t를 통해 -1, 0, +1 구간으로 나눈다. 이후 -1과 +1에 각각 -Scale과 +Scale 값을 곱하여 근사 값을 구하는 방법이다. 학습할 때에는 -Scale과 +Scale을 학습한다.

[3]의 방법은 성능 하락을 막기 위해 신경망의 첫 번째 층과 마지막 층은 32-Bit 실수 연산을 하고 나머지 층에서는 2-Bit 연산을 한다. 하지만 2-Bit 연산이 지원되는 CPU는 많지 않으며 연산을 가속화하기 어렵다. 2-Bit 연산을 가속화하기 위해서는 Field Programmable Gate Array (FPGA)를 통해 구현하는 것이 일반적이다.

3. 1-Bit(이진) 신경망

1-Bit 신경망은 논리 연산인 XNOR 연산을 통해 곱셈을 구현할 수 있으며 레지스터 내의 1로 설정된 비트의 수를 알 수 있는 Pop-Count 명령을 통해 누적 덧셈을 구현할 수 있다. 따라서 실수나 정수 곱셈, 덧셈이 필요하지 않게 됨으로 연산 속도가 증가한다. 또한 기존 32-Bit에서 1-Bit로 줄어들기 때문에 이론적으로 메모리 대역폭이 32배 증가하게 된다.

1-Bit 신경망은 [1]의 방법이 있다. [1]의 방법은 입력과 가중치를 모두 1-Bit로 변환 후 XNOR 연산을 한다. XNOR 연산 결과에 근사 값을 곱하여 32-Bit에서 1-Bit로의 변환으로 인한 손실을 보정한다.

[1]에서는 [3]과 같이 신경망의 첫 번째 층과 마지막 층은 32-Bit 실수 연산을 통해 성능 하락을 막으려 했지만 [3]과는 달리 성능 하락이 10% 정도로 크게 나타났다.

III. 제안하는 방법

1. 1-Bit 연산 방법

신경망 연산은 곱셈과 덧셈으로 이루어져 있기 때문에 1-Bit 곱셈과 누적 덧셈을 구현하는 방법으로 XNOR 논리 연산을 사용할 수 있다. XNOR의 진리표는 표 1과 같다. 비트 표현에서 이진수 1은 +1이며 이진수 0은 −1로 표현된다.

Table 1. XNOR Truth Table.
표 1. XNOR 진리표

Input		Output
A	B	A xnor B
0	0	1
0	1	0
1	0	0
1	1	1

곱셈 후 누적 덧셈을 구현하기 위해서는 Pop-Count 연산을 사용하여 구현이 가능하다. Pop-Count 연산은 표 2와 같이 비트 내에서 1로 설정된 비트의 개수를 반환한다. Pop-Count 연산의 결과에 2를 곱하고 총 비트 수를 빼면 누적 덧셈이 가능하다.

Table 2. Accumulation after multiplication.
표 2. 곱셈 후 누적 덧셈

8-Bit Register A	1011 0100
8-Bit Register B	0110 1101
A xnor B	0010 0110
PC(A xnor B)	3
2 * PC(A xnor B) - 8	-2

신경망의 파라미터를 수식 6과 같이 이진화 후 N개의 파라미터를 N-Bit 레지스터에 Packing하여 N개의 곱셈을 한 번의 XNOR 논리 연산을 통해 가속화할 수 있다.

$$\gamma_b = \begin{cases} +1, \text{if } \gamma \geq 0, \\ -1, otherwise \end{cases} \quad (6)$$

신경망의 파라미터를 수식 6과 같이 이진화 후 N개의 파라미터를 N-Bit 레지스터에 Packing하여 N개의 곱셈을 한 번의 XNOR 논리 연산을 통해 가속화한다. 표 3은 N-Bit-Packing 알고리즘을 설명한다. 3번 줄은 파라미터가 0보다 크면 1 아니면 0을 의미한다. 4번 줄에서 ≪는 비트를 왼쪽으로 밀고 맨 오른쪽 비트를 0으로 채우는 연산이다. 따라서 $sign$ 변수의 비트를 왼쪽으로 q번 밀어서 v 변수와 OR 연산하여 v 변수에 저장하는 것을 의미한다.

Table 3. N-Bit-Packing.
표 3. N-Bit-Packing

Algorithm	N-Bit-Packing
입력 : N 개의 실수 배열 r	
출력 : Bit-Packing된 N-Bit 레지스터 v	
1	$v = 0$
2	***for*** $q = 0$ ***to*** $N-1$ ***do***
3	$sign = r[q] \geq 0 \,?\, 1 : 0$
4	$v = v \mid (sign \ll q)$

합성 곱 연산을 구현하는 방법은 다양한 방법[4, 5]이 존재하는데 본 논문에서는 그림 1과 같은 방법을 사용하여 행렬 곱이 가능한 형태로 변환하는 방법을 사용한다.

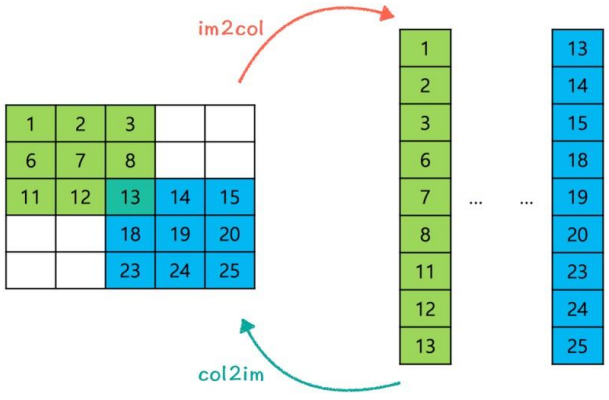

Fig. 1. Patch Transform.
그림 1. 패치 변환

입력이 $N \times C \times H \times W$(N : 연산할 입력의 개수, C : 입력의 채널, H : 입력의 세로 크기, W : 입력의 가로 크기, 예: 가로 32, 세로 32 인 RGB 영상이 5개일 때 $5 \times 3 \times 32 \times 32$)형태이고 가중치가 $C \times Q \times K \times K$ (C : 입력의 채널, Q : 출력 채널, K : 가중치 크기) 일 때, 가중치를 S의 크기만큼 이동하는 합성 곱 연산을 그림 1의 알고리즘으로 변형하면 입력 I^n는 가로가 I_c이고 세로가 I_r인 행렬이 된다. 가중치

W는 가로가 W_c이고 세로가 W_r인 행렬이 된다. I_c와 I_r는 수식 7과 같고 W_c와 W_r는 수식 8과 같다.

$$I_r = K \times K \times C$$
$$I_c = \left\lfloor \frac{(H-K+1)}{S}+1 \right\rfloor \left\lfloor \frac{(W-K+1)}{S}+1 \right\rfloor \quad (7)$$

$$W_r = Q$$
$$W_c = K \times K \times C \quad (8)$$

따라서 그림 1을 통해 입력과 가중치를 변형하여 수식 9와 같이 입력의 개수만큼 N번 행렬 곱을 진행하면 합성 곱 연산을 할 수 있게 된다.

$$U^n = W \times I^n \quad (9)$$

합성 곱 연산을 행렬 곱으로 표현이 가능하다는 것을 보였기 때문에 입력과 가중치가 1-Bit로 표현되고 이를 P-Bit-Packing 하였을 때 표 4와 같이 1-Bit 행렬 곱 알고리즘을 통해 연산 가속화가 가능하다.

Table 4. 1-Bit Matrix Multiplication.
표 4. 1-Bit 행렬 곱

Algorithm	1-Bit Matrix Multiplication
Input : Mat A, Mat B, Col of A M, Row of A K, Col of B N	
Output : Mat C	
1	*for* $r=0$ *to* $M-1$ *do*
2	*for* $c=0$ *to* $N-1$ *do*
3	$C[r][c]=0$
4	$sum=0$
5	*for* $k=0$ *to* $K-1$ *do*
6	$sum \mathrel{+}= PopCount(Xnor(A[r][k],B[k][r]))$
7	$C[r][c] \mathrel{+}= 2 \times sum - P \times K$

가로와 세로의 크기가 $N \times N$인 행렬 곱셈은 $O(N^3)$의 시간 복잡도를 갖는다. 표 4의 1-Bit 행렬 곱셈은 M, K, N이 모두 같을 때 $O(\frac{N^3}{P})$의 시간 복잡도이며 실수 곱셈이 아닌 XNOR 연산과 Pop-Count 연산을 사용하기 때문에 연산 속도가 더 빠르고 P-Bit-Packing이 되었기 때문에 메모리 접근 횟수도 P배 줄어들게 된다.

2. 합성 곱 근삿값

[1]에서는 근삿값을 구할 때 그림 2와 같이 각 특징 맵에 근삿값을 적용하였다. 이러한 방법은 각 필터마다 절댓값 평균을 계산해야하기 때문에 연산 가속화가 더디게 된다. 그림 2를 수식화 하면 수식 10과 같다.

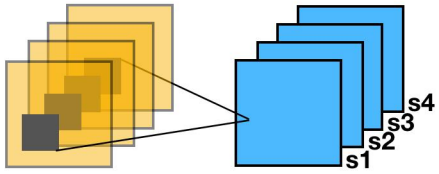

Fig. 2. Existing Approximation.
그림 2. 기존의 근삿값 적용

$$s_q^l = \frac{1}{C \times K_2 \times K_1} \sum_{c=1}^{C} \sum_{k_2=1}^{K_2} \sum_{k_1=1}^{K_1} \left| w_{q,c,k_2,k_1} \right| \quad (10)$$

본 논문에서는 근삿값을 각 필터마다 절댓값 평균을 구하는 것이 아닌 필터 전체에 대해 절댓값 평균을 구하여 모든 특징 맵에 적용하는 방법을 사용하였다. 이러한 방법은 그림 3과 같고 수식으로 표현하면 수식 11과 같다.

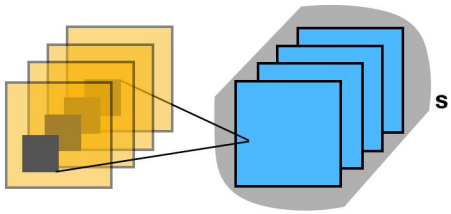

Fig. 3. Proposed Approximation.
그림 3. 제안하는 근삿값 적용

$$s^l = \frac{1}{Q \times C \times K_2 \times K_1} \sum_{q=1}^{Q} \sum_{c=1}^{C} \sum_{k_2=1}^{K_2} \sum_{k_1=1}^{K_1} \left| w_{q,c,k_2,k_1} \right| \quad (11)$$

3. 비선형 활성화 함수 제거

1-Bit 연산에서 입력을 -1 또는 $+1$ 로 양자화하는 것이 그림 4와 같이 계단 함수를 사용하는 효과를 갖기 때문에 [6]과 같은 비선형 활성화 함수를 없앨 수 있다.

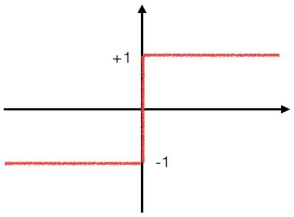

Fig. 4. Step Function.
그림 4. 계단 함수

신경망의 다음 층의 입력으로 전달할 때에는 수식 12과 같이 선형 함수를 사용한다. 입력에 대한 미분 값인 $\frac{\partial y}{\partial x^b}$은 수식 13과 같이 출력 층으로부터 계산된 미분 값을 그대로 전달한다.

$$y = f(x^b) = x^b \\ , where\ f(\gamma) = \gamma \tag{12}$$

$$\frac{\partial y}{\partial x^b} = \frac{\partial E}{\partial y} \cdot 1 \tag{13}$$

하지만 $sign(\cdot)$ 함수의 미분은 가능하지 않으며 또한 입력 값이 너무 크면 성능 하락의 요인이 되기 때문에 [7]에 의한 방법으로 수식 13과 같이 입력이 -1와 $+1$ 사이이면 출력 층의 미분 값을 그대로 전달하고 그 외에는 전달하지 않는다. $1_{|x| \le 1}$을 그림으로 표현하면 그림 5와 같다.

$$x^b = sign(x) \tag{14}$$

$$\frac{\partial x^b}{\partial x} = \frac{\partial E}{\partial x^b} \cdot 1_{|x| \le 1} \tag{15}$$

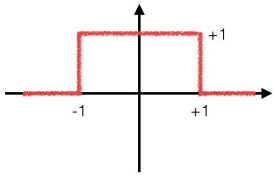

Fig. 5. Sign Function Gradient.
그림 5. Sign 함수 미분

4. 1-Bit 신경망 성능 하락 완화

[1]의 방법은 첫 번째 층과 마지막 층을 32-Bit 연산을 적용한다. 하지만 첫 번째 층의 합성 곱 연산 출력을 이진화하면 특징 표현력이 매우 떨어지는 것을 실험적으로 확인하였다.

본 논문에서는 첫 번째 층의 합성 곱 연산뿐만 아니라 두 번째 층의 합성 곱 연산까지 32-Bit 연산으로 변경하였고 성능이 크게 상승되었음을 실험적으로 확인하였다.

32-Bit 연산이 적용되는 곳에는 [6]과 같은 비선형 활성화 함수를 함께 사용하였다.

Ⅳ. 실험

본 논문에서 제안한 방법을 검증하기 위해 [8]의 방법을 사용하여 차량 번호판 검출 신경망을 구현하였다. 표 5는 본 논문에서 사용할 차량 번호판 검출 신경망으로 1번, 3번, 15번, 16번, 22번, 23번 층에 32-Bit 합성 곱 연산을 적용했다. 그 외의 8개의 합성 곱 연산은 입력과 가중치가 1-Bit 연산으로 적용된다.

학습할 때에는 기존 2,000장의 데이터를 잡음 및 컬러 등의 변형을 주어 대략 8,000장 정도로 증대(Augmentation)하였다. 차량 번호판 데이터는 그림 6와 같이 구형, 신형 번호판과 주/야, 날씨 등 다양한 환경에서 촬영된 영상으로 구성되어 있다.

Fig. 6. Vehicle License Plate Data.
그림 6. 번호판 데이터

Table. 5 Vehicle License Plate Detection Neural Network.
표 5. 번호판 검출 신경망

No.	Type	Filters	Size / Stride	Output
1	Conv	16	3×3 / 1	320×320
2	Max Pool		2×2 / 2	160×160
3	Conv	32	3×3 / 1	160×160
4	Max Pool		2×2 / 2	80×80
5	Conv	64	3×3 / 1	80×80
6	Max Pool		2×2 / 2	40×40
7	Conv	128	3×3 / 1	40×40

No.	Type	Filters	Size / Stride	Output
8	Max Pool		2×2 / 2	20×20
9	Conv	256	3×3 / 1	20×20
10	Conv	128	1×1 / 1	20×20
11	Conv	256	3×3 / 1	20×20
12	Max Pool		2×2 / 2	10×10
13	Conv	512	3×3 / 1	10×10
14	Conv	256	1×1 / 1	10×10
15	Conv	512	3×3 / 1	10×10
16	Conv	24	1×1 / 1	10×10
17	Anchor	84,35, 107,42, 77,59, 148,39		
18	Route (13)			
19	Up Sample	512		20×20
20	Route (19, 9)			20×20
21	Conv	256	1×1 / 1	20×20
22	Conv	512	3×3 / 1	20×20
23	Conv	24	1×1 / 1	20×20
24	Anchor	35×22, 49,17, 72,20, 59,38		

경사 하강법(Gradient Descent)으로 가중치를 갱신하였고 Momentum은 0.9, 가중치 Decay는 0.0005, 학습률은 0.001이고 총 학습 횟수는 70,000번이다. 학습 횟수 60,000번 이후에는 0.1배 하였다. 학습 데이터의 미니배치 크기는 64로 정하였다. 가중치를 학습하는 연산(예, 합성 곱)에는 모두 배치 정규화[9] 기법을 적용하였다. [1]의 방법은 신경망의 수렴이 3~4배 더 느려지며 대부분의 영상에서 검출이 안 되었다.

그림 7은 [1]의 방법과 본 논문에서 제시한 방법의 결과이다. 첫 번째와 두 번째 열의 [1]의 방법은 검출의 위치는 정확하지만 Intersection over Union (IoU)이 낮다. 반면의 제안 방법의 결과는 IoU가 높은 것을 볼 수 있다.

세 번째, 네 번째 열은 [1]의 방법으로 검출이 불가능한 영상이다. 몇 영상은 검출이 되지 않았으며 일반화 성능이 떨어지는 것을 실험적으로 확인하였다. 반면에 제안된 방법은 검출 위치와 IoU가 정확하다.

그림 8의 왼쪽은 표 5번의 신경망을 32-Bit 연산으로 학습된 결과이고 오른쪽은 본 논문에서 제안한 방법이 적용된 결과이다.

그림 9는 표 5에 대해 왼쪽은 [1]의 방법과 제안된 방법의 속도 비교이고 오른쪽은 정확도 비교이다. 그림 10의 왼쪽은 표 5에 대해 [1] 방법과 제안된 방법의 Multiplier-Accumulator (MAC) 비교이다. 연산 횟수가 18% 늘어났지만 32-Bit와의 비교는 95%의 연산 횟수를 줄일 수 있다.

그림 9의 왼쪽은 늘어난 18% 연산 횟수와 비슷하게 총 연산 지연이 [1]의 방법인 153ms에서 제안된 방법인 212ms로 늘어났다. 측정된 시간은 Intel i7-7700K @4.20GHz CPU에서 측정하였으며 최적화되지 않은 코드에서 실행되었다. 그림 9의 오른쪽에서 [1]의 방법에 의해 학습된 차량 번호판 검

Fig. 7. Left : results of previous method,
Right : results of our method.
그림 7. 왼쪽 : 이전 연구의 결과, 오른쪽 : 본 연구의 결과

Fig. 8. Left : 32-Bit Neural Net. Results,
Right : Proposed 1-Bit Neural Net. Results
그림 8. 왼쪽 : 32-Bit 신경망 결과,
오른쪽 : 제안된 1-Bit 신경망 결과

출기의 성능은 74%의 정확도를 얻은 반면에 본 연구에서 제안한 방법으로는 96.1%로 상승하였다. 32-Bit와 비교해서는 정확도가 3.2% 저하되었다.

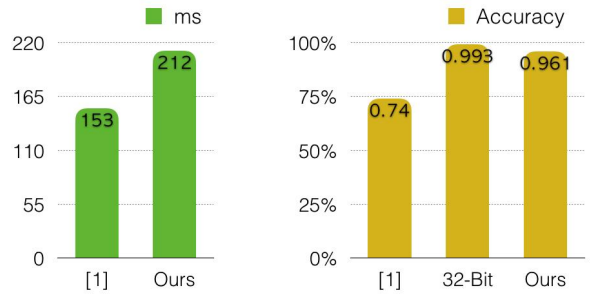

Fig. 9. Left: Compare previous method with ours, Right : Compare accuracy on previous method with ours.
그림 9. 왼쪽 : 기존 연구와의 속도 비교,
오른쪽 : 기존 연구와의 정확도 비교

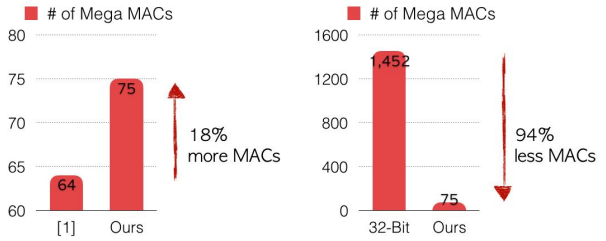

Fig. 10. Left : Compare previous method with ours number of MACs, Right : Compare 32-bit network with ours number of MACs.
그림 10. 왼쪽 : 이전 연구와의 MACs 수 비교,
오른쪽 : 32-bit 신경망과의 MACs 수 비교

V. 결론 및 향후 연구

본 논문에서는 기존의 1-Bit 신경망의 성능 하락을 완화하기 위한 방법을 제안한다. 제안된 방법을 검증하기 위해 객체 검출 신경망을 이용하여 실험하였으며 기존 방법의 결과인 74%의 정확보다 상승된 96.1%의 정확도를 얻었다.

향후 연구에서는 배치 정규화와 경사 하강법의 적용 방법을 변경하고 미분 값을 양자화하여 학습 속도를 높이는 방안과 제안된 성능보다 높은 성능을 얻을 수 있는 양자화 신경망을 연구할 것이다.

References

[1] M. Rastegari, V. Ordonez, J. Redmon and A. Farhadi., "Xnor-net: Imagenet classification using binary convolutional neural networks," *European Conference on Computer Vision*, pp.525-542, 2016.

[2] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *Computer Vision and Pattern Recognition* pp.2704-2713, 2018.

[3] C. Zhu, S. Han, H. Mao and W. J. Dally, "Trained ternary quantization," *International Conference on Learning Representations*, 2017.

[4] Michae¨l Mathieu, Mikael Henaff and Yann LeCun, "Fast training of convolutional networks through ffts," *International Conference on Learning Representations*, 2013.

5] A. Lavin and S. Gray, “Fast algorithms for convolutional neural networks,” *Computer Vision and Pattern Recognition*, pp.4013-4021, 2016.
[6] K. He, X. Zhang, S. Ren and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” *International Conference on Computer Vision*, pp.1026-1034, 2015.
[7] Y. Bengio, N. Léonard and A. Courville, “Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation,” https://arxiv.org/abs/1308.3432.
[8] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” *Computer Vision and Pattern Recognition,* pp.6517-6525, 2017.
[9] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, *International Conference on International Conference on Machine Learning*, pp.448-456, 2015.

BIOGRAPHY

Sung-hoon Im (Student Member)

2017 : BS degree in Computer Engineering, Hanbat National University.
2017~Now : MS degree course in Computer Engineering, Hanbat National University.

Jae-heung Lee (Member)

1983 : BS degree in Electronic Engineering, Hanyang University.
1985 : MS degree in Electronic Engineering, Hanyang University.
1994 : PhD degree in Electronic Engineering, Hanyang University.
1989~Now : Professor in Dept. of Computer Engineering, Hanbat National University.