# 딥러닝 네트워크 압축을 위한 양자화 오프셋의 바이어스 임베딩 기법

*정진우 **김성제 ***홍민수
한국전자기술연구원
*jw.jeong@keti.re.kr

# Bias embedding of quantization offset for convolutional network compression

*Jinwoo Jeong **Sungjei Kim ***Minsoo Hong
Korea Electronics Technology Institute

## 요약

본 논문은 딥러닝 네트워크의 압축을 위한 양자화 오프셋의 바이어스 기법을 제안한다. 양자화는 32비트 정밀도를 갖는 가중치와 활성화 데이터를 특정 비트 이하의 정수로 압축한다. 양자화는 원 데이터에 스케일과 오프셋을 더함으로써 수행되므로 오프셋을 위한 합성곱 연산이 추가된다. 본 논문에서는 입력 활성화 데이터의 양자화 오프셋과 가중치의 합성곱의 출력은 바이어스에 임베딩될 수 있음을 보여준다. 이를 통해 추론 과정 중 오프셋의 합성곱 연산을 제거할 수 있다. 실험 결과는 오프셋의 합성곱이 바이어스에 임베딩이 되더라도 영상 분류 정확도에 영향이 거의 없음을 증명한다.

## 1. 서론

최근 딥러닝 기술의 확산이 가속화 되고 있어 딥러닝 기술을 하드웨어, 즉 SoC로 구현하기 위한 연구들이 많이 진행되고 있다. 매우 많은 파라미터와 연산량을 가진 기존 딥러닝 네트워크를 하드웨어로 구현하려면 많은 비용과 전력 소모가 요구되므로 네트워크를 효과적으로 줄이기 위한 여러 가지 압축 방법 들이 제안되고 있다. 대표적인 압축 방법으로는 지식 증류 (Knolwedge distillation) [1], 네트워크 가지치기 (Network pruning), 양자화 (Quantization), 엔트로피 코딩 (Entropy Coding) [2] 등이 있다. 본 논문에서는 이 중 양자화 기법을 적용할 시 효과적으로 합성곱 연산의 일부분을 제거할 수 있는 방법에 대하여 제안한다.

## 2. 네트워크 양자화 기법

딥러닝 네트워크의 가중치와 활성화 데이터는 32비트의 부동 소수점 정밀도를 갖는다. 하드웨어로 구현할 경우 부동 소수점 연산은 많은 게이트 수와 전력량을 요구함으로 이를 적절한 정밀도를 갖는 정수형 자료형으로 변환할 필요가 있다. 이를 양자화 (Quantization)라고 하며 일반적으로는 8비트 이하의 정밀도로 양자화를 수행한다.

양자화는 식 (1) 과 같이 표현할 수 있다. $X_f$는 32비트 정밀도를 갔는 데이터이고, $S_X$는 양자화 스케일로 부동 소수점 형식의 자료형이다. $Z_X$ 는 양자화된 신호가 0에서 $2^B-1$의 사이에 존재하게 하는 오프셋으로 정수이다. $B$는 양자화에 의한 비트 정밀도를 의미하며 8일 경우 $X_q$는 0에서 255 사이의 정수이다. $\lfloor X \rceil$ 는 $X$에 대한 반올림 함수이다.

$$X_q = \lfloor S_X X_f - Z_X \rceil \quad (1)$$

양자화된 신호에 의한 합성곱(Covolutional) 계층의 연산은 다음과 같이 표현할 수 있다. 식 (1)은 입력 신호의 양자화를 의미하고 식 (2)와 (3)은 가중치 (Weight)와 바이어스 (Bias) 의 양자화라고 하면 식(4)는 합성곱 계층의 양자화 결과이다. $*$ 은 합성곱 연산을 의미한다.

$$W_q = \lfloor S_W W_f - Z_W \rceil \quad (2)$$

$$B_q = \lfloor S_B B_f - Z_B \rceil \quad (3)$$

$$\begin{aligned} Y_q &= \lfloor S_Y(X_f * W_f + B_f) - Z_Y \rceil \\ &= \left\lfloor \frac{S_Y}{S_X S_W}\left\{(X_q + Z_X) * (W_q + Z_W) + \frac{S_X S_W}{S_B}(B_q + Z_B)\right\} - Z_Y \right\rfloor \end{aligned} \quad (4)$$

합성곱 연산의 입력은 $X_q + Z_X$와 와 $W_q + Z_W$이 된다. 그런데 이 입력들의 정밀도 범위는 오프셋이 더해짐에 따라 기존의 $B$ 비트 범위를 보장할 수 없게 된다. 만약 합성곱기에 $X_q$와 $W_q$를 입력으로 넣으면 오프셋으로 인한 부가 연산으로 $Z_X * W_q + X_q * Z_W + Z_X * Z_W$ 의 연산이 필요하다. 합성곱 연산의 입력을 $B$ 비트 범위로 제한하고 부가적인 연산을 피하기 위해 본 논문에서는 오프셋의 합성곱 연산을 바이어스에 임베딩시키는 방법을 제안한다.

## 3. 양자화 오프셋의 바이어스 임베딩 기법

가중치의 경우 0을 중심으로 분포하는 경우가 일반적이기 때문에 계산의 단순성을 위해 본 논문에서는 $Z_W$를 0이며 대칭적으로 양자화한다고 가정하면 이 경우 양자화된 가중치의 범위는 $-2^{B-1}$에서 $2^{B-1}-1$ 사이이다. 이 겨우 식 (4)는 식(5)와 같이 표현할 수 있다.

$$Y_q = \left\lfloor \frac{S_Y}{S_X S_W}\left\{(X_q + Z_X) * W_q + \frac{S_X S_W}{S_B}(B_q + Z_B)\right\} - Z_Y \right\rfloor \quad (5)$$

식 (5)에서 양자화 되기 전의 합성곱의 수식은 식(6)과 같이 표현되며 $Z_X * W_q$를 바이어스에 임베딩하도록 한다.

$$Y_{accum} = X_q * W_q + Z_X * W_q + \frac{S_X S_W}{S_B}(B_q + Z_B) \quad (6)$$

$Z_X$ 는 입력 활성화 (activation) 데이터의 오프셋으로 $X_q$와 동일한 데이터 크기 ($N \times H \times W$)를 갖는 3차원 데이터이다. $N$, $H$, $W$는 각각 $X_q$의 채널, 수직, 수평 크기를 의미하며, 활성화 데이터의 경우 3차원 데이터는 동일한 값인 $z_X$ 로 구성된다. $W_q$ 는 4차원 데이터로 $M \times N \times K \times K$의 크기를 갖으며 $M$은 3차원 필터의 개수를 $K$는 2D 커널 크기를 나타낸다. $Y_{accum}$은 합성곱의 스트라이드가 1이며 패딩이 $(K-1)$만큼 되었다고 가정하면 가정하면 $M \times H \times W$의 데이터 크기를 갖는다. 바이어스 부분인 $S_X S_W / S_B(B_q + Z_B)$는 상수 값으로 구성되어 있으므로 추론 (Inference)을 수행하기 전에 미리 계산해 놓도록 할 수 있다. 따라서 추론에 사용되는 실제 수식은 식 (7)과 같다.

$$Y_{accum} = X_q * W_q + Z_X * W_q + B_{accum} \quad (7)$$

$Z_{accum}(m)$은 $Z_X$와 $W_q$의 합성곱의 $m$번째 채널의 출력을 나타내며 식 (8)과 같이 표현할 수 있다. ★는 교차상관함수 (cross correlation)을 의미한다. 출력의 $m$번째 채널에서 $< x, y >$ 위치의 결과는 식 (9)와 같이 나타낼 수 있다. 위에서 서술하였듯이 $Z_X$는 모든 위치에 대하여 동일한 값인 $z_X$를 가지므로 식 (9)는 식 (10)과 같이 변형되어 진다.

$$Z_{accum}(m) = Z_X * W_q(m) = \sum_{n=1}^{N} W_q(m,n) \star Z_X(n) \quad (8)$$

$$Z_{accum}(m)_{<x,y>} = \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K} W_q(m,n,i,j) Z_X(c, y+j-1, x+i-1) \quad (9)$$

$$Z_{accum}(m)_{<x,y>} = z_X \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{K} W_q(m,n,i,j) \quad (10)$$

식 (10)에서 $m$번째 채널에서 $< x, y >$ 와 상관 없이 $Z_{accum}(m)$은 모두 동일한 값을 가짐을 알 수 있다. 따라서 식 (10)은 식 (11)과 같이 요약될 수 있으며 이는 $Z_{accum}$은 바이어스와 같이 채널별로 하나의 값만 갖는다는 것을 의마한다. 따라서 $Z_{accum}$을 바이어스에 더하여 새로운 바이어스를 식 (12)와 같이 정의할 수 있다.

$$Z_{accum}(m) = z_X \times \sum_{all} W_q(m) \quad (11)$$

$$B_{Fmb}(m) = z_{acuum}(m) + B_{acuum}(m) \quad (12)$$

최종적으로 추론 과정에서는 식 (13)과 같이 임베딩된 바이어스인 $B_{Fmb}$를 사용함으로써 합성곱 입력의 비트 정밀도를 제한하는 동시에 $Z_{accum}$을 산출하기 위한 합성곱 연산을 제거할 수 있다.

$$Y_{accum} = X_q * W_q + B_{Fmb} \quad (13)$$

## 4. 실험 결과 및 결론

제안 알고리즘의 성능 평가를 위해 우분투 18.04. Pytorch1.4.0에서 Nvidia RTX Titan을 사용하여 테스트하였다. 실험은 EfficientNet-B0 [3] 에 대하여 바이어스 임베딩을 한 결과와 그렇지 않은 결과를 비교하였다. 양자화는 학습후 양자화 기법 (Post Train Quantization) 기법을 사용하여 학습된 가중치에 대하여 양자화를 수행하였으면 가중치와 활성화 데이터에 대하여 각각 양자화를 수행하였다. EfficientNet-B0에서 활성화 데이터는 Swish 대신에 Relu6를 사용하였으며 실험 데이터로는 Imagenet과 Cifar10 데이터를 사용하였다.

표1 과 표2는 Imagenet과 Cifar10에 대한 결과를 각각 보여준다. 활성화 데이터와 가중치는 각각 8과 6비트로 양자화 되었다. 실험 결과는 바이어스 임베딩 전과 임베딩 후의 결과가 거의 유사함을 확인할 수 있다. 완벽하게 동일한 결과 나오지 않는 것은 영상의 경계 부분을 0으로 패딩함으로써 식(10)은 영상의 경계부분에서는 성립하지 않는다. Imagenet과 같이 입력 영상이 224x224로 클 경우 영상 경계 부분이 미치는 영향은 작은데 반하여 Cifar10의 경우 입력이 32x32 영상이므로 경계에서의 작은 차이에도 영향을 받는다.

실험결과는 본 논문에서는 입력 활성화 데이터의 양자화 오프셋과 가중치의 합성곱의 출력은 바이어스에 임베딩될 수 있음을 보여준다.

<표 1> Imagenet 에 대한 제안 방법의 성능 평가, 양자화 전 Top1 분류 정확도: 76.754

| 비트 정밀도 | | 분류 (classification) Top1 정확도 | |
|---|---|---|---|
| 활성화 | 가중치 | 바이어스 임베딩 전 | 바이어스 임베딩 후 |
| 8 | 8 | 76.53 | 76.55 |
| 8 | 6 | 75.45 | 75.41 |
| 6 | 8 | 74.90 | 75.01 |
| 6 | 6 | 73.12 | 73.26 |
| 평균 | | 75.00 | 75.06 |

<표 2> Cifar10 에 대한 제안 방법의 성능 평가, 양자화 전 Top1 분류 정확도: 91.77

| 비트 정밀도 | | 분류 (classification) Top1 정확도 | |
|---|---|---|---|
| 활성화 | 가중치 | 바이어스 임베딩 전 | 바이어스 임베딩 후 |
| 8 | 8 | 91.65 | 91.61 |
| 8 | 6 | 91.27 | 91.20 |
| 6 | 8 | 90.67 | 90.53 |
| 6 | 6 | 90.09 | 90.14 |
| 평균 | | 90.92 | 90.87 |

## Acknowledgements

이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2019-0-01351, 활성화/커널데이터의 압축/복원을 통한 초저전력 모바일 딥러닝 반도체 기술 개발)

## References

[1] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." NIPS Deep Learning and Representation Learning Workshop (2015)

[2] Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

[3] Mingxing Tan, Quoc V. Le: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019: 6105-6114