The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 21, No. 1, pp.93-98, Feb. 28, 2021. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2021.21.1.93
JIIBC 2021-1-13

양자화 기반의 모델 압축을 이용한 ONNX 경량화

Lightweight of ONNX using Quantization-based Model Compression

장두혁*, 이정수, 허준영*****

Duhyeuk Chang*, Jungsoo Lee**, Junyoung Heo***

요 약 딥 러닝의 발전으로 다양한 AI 기반의 응용이 많아지고, 그 모델의 규모도 매우 커지고 있다. 그러나 임베디드 기기와 같이 자원이 제한적인 환경에서는 모델의 적용이 어렵거나 전력 부족 등의 문제가 존재한다. 이를 해결하기 위해서 클라우드 기술 또는 오프로딩 기술을 활용하거나, 모델의 매개변수 개수를 줄이거나 계산을 최적화하는 등의 경량화 방법이 제안되었다. 본 논문에서는 다양한 프레임워크들의 상호 교환 포맷으로 사용되고 있는 ONNX(개방형 신경망 교환 포맷) 포맷에 딥러닝 경량화 방법 중 학습된 모델의 양자화를 적용한다. 경량화 전 모델과의 신경망 구조와 추론 성능을 비교하고, 양자화를 위한 다양한 모듈 방식를 분석한다. 실험을 통해 ONNX의 양자화 결과, 정확도는 차이가 거의 없으며 기존 모델보다 매개변수 크기가 압축되었으며 추론 시간 또한 전보다 최적화되었음을 알 수 있었다.

Abstract Due to the development of deep learning and AI, the scale of the model has grown, and it has been integrated into other fields to blend into our lives. However, in environments with limited resources such as embedded devices, it is exist difficult to apply the model and problems such as power shortages. To solve this, lightweight methods such as clouding or offloading technologies, reducing the number of parameters in the model, or optimising calculations are proposed. In this paper, quantization of learned models is applied to ONNX models used in various framework interchange formats, neural network structure and inference performance are compared with existing models, and various module methods for quantization are analyzed. Experiments show that the size of weight parameter is compressed and the inference time is more optimized than before compared to the original model.

Key Words : CIFAR-10, Deep learning, MNIST, Memory Useage, ONNX, ONNXRuntime, Quantization, TFLite

Ⅰ. 서 론

1. 연구내용

딥 러닝의 발전으로 AI에 관한 관심이 늘어나며 연구가 활발히 진행됨에 따라 다양한 방법론과 기술이 등장하고 있다. 또한, 해당 기술들을 실제 환경에 적용하기 위한 연구도 진행되고 있으며, 실제로 객체 탐지나 음성 인식 등의 분야로 사용되고 있다. 해당 기술들의 성능을 향

*학생회원, 한성대학교 컴퓨터공학부
**학생회원, 한성대학교 컴퓨터공학과
***정회원, 한성대학교 컴퓨터공학부(교신저자)
접수일자 2020년 12월 5일, 수정완료 2021년 1월 5일
게재확정일자 2021년 2월 5일

Received: 5 December, 2020 / Revised: 5 January, 2021 / Accepted: 5 February, 2021
*Corresponding Author: jyheo@hansung.ac.kr
Division of Computer Engineering, Hansung University, Korea

상하는 가장 단순한 방법은 딥러닝 네트워크를 더 깊고 넓게 쌓아 올리는 것이지만, 이는 더 많은 연산 처리와 전력을 필요로 한다. 고성능 GPU를 갖춘 서버 컴퓨터에서는 이런 대규모 모델의 처리가 어느정도 가능하지만, 이런 모델들을 모바일 등의 임베디드 기기에서 직접 처리하는 것은 리소스나 배터리 등의 문제로 인해 실현되기 어렵다[7][8].

이러한 단점을 극복하기 위해 이미지 데이터 압축하는 방법과 모델 압축 또는 경량화 기법이 등장하였다. 이는 다양한 방법을 적용해 딥 러닝 모델의 리소스 사용량, 특히 신경망의 매개변수 개수를 줄인 후[10] 시스템에 적용하는 방법이다.

본 논문에서는 Microsoft에서 개발한 오픈 신경망 교환 포맷(ONNX, Open Neural Network Exchange)[9]에 경량화 방법 중 양자화(Quantization)를 적용하고 모델 크기, 네트워크 구조 변화, 추론 결과 차이 등 최적화 전후 결과를 비교한다. 또한, 이후 다양한 프레임워크들의 최적화 함수를 적용하기 위해 각 프레임워크가 지원하는 모델로 변환하는 방법을 설명한다.

본 논문의 구성은 다음과 같다. 2장에서는 ONNX 라이브러리의 모델 포맷과 변환 과정 및 추론 엔진을 설명한다. 3장에서는 양자화 중 학습 후 양자화(Post Training Quantization) 방법과 이를 ONNX 라이브러리에서 적용하는 방법을 설명한다. 4장에서는 기존 모델과 양자화가 적용된 모델의 추론 시간 비교 및 ONNX 라이브러리의 양자화 결과와 TFLite 라이브러리의 양자화 결과를 비교한다. 마지막으로 5장에서 결론과 향후 연구방향을 제시한다.

II. ONNX

1. ONNX 포맷

오픈 신경망 교환 포맷[2][3] (ONNX, Open Neural Network Exchange)은 Microsoft, Facebook에서 오픈소스 프로젝트로 개발한 딥러닝 모델의 표현 방식으로, 다양한 인공지능 프레임워크 간의 상호 호환성을 위해 개발되었다. ONNX는 Tensorflow, PyTorch, Caffe등 다수의 딥러닝 프레임워크에서 지원되고 있으며, 내부적으로는 protobuf 라는 데이터 구조를 바탕으로 이루어져 있다. 또한, 제어 흐름을 통한 표준화된 그래프 표현 형식으로 나타내어지기 때문에, 여러 프레임워크에서 학습한 모델을 ONNX 모델 포맷으로 변환하기가 쉽다. ONNX는 자체 연산자를 가지는데, 다음 표1은 ONNX 연산자 중 일부에 대한 예시이며, 다음과 같은 임무를 수행한다.

표 1. 실험 파라미터
Table 1. Experiment parameter

연산자명	역할
Add	Performs element-wise binary addition
MatMul	Performs Matrix product
Cast	Casts the elements of a given input tensor to a specified data type
Div	Performs element-wise binary division

2. 딥 러닝 모델을 ONNX로 변환

본 실험에 사용된 ONNX 모델의 원시 모델은 대표적인 딥러닝 프레임워크 중 하나인 Keras를 통해 생성하여, 해당 프레임워크로 저장한 .h5 모델을 keras2onnx 라이브러리의 API를 사용하여 ONNX 모델로 변환하였다. 이 과정에서 기존 Keras 모델 그래프의 일부분은 ONNX 연산자로 표현되며, 그림1은 해당 변환 전 후의 모습이다.

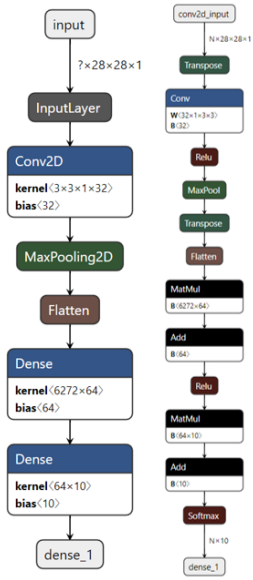

그림 1. CNN 모델의 ONNX 변환 결과
Fig. 1. Result of CNN model converted to ONNX

3. ONNX런타임

ONNX런타임은 ONNX 모델 포맷에 대해 직접 추론 기능을 수행할 수 있는 ONNX에서 제공하는 자체 추론 엔진 모듈이다. AI 시스템 개발자들은 기존 시스템에 맞

줘 이 런타임을 최적화 및 통합할 수 있고, 여러 운영체제 환경에서 컴파일 및 빌드할 수 있다. 본 논문에서는 Keras의 .h5 모델의 자체 엔진을 이용한 추론 결과와 변환된 ONNX 모델의 ONNX런타임 엔진을 이용한 추론 결과를 비교하였다.

Ⅲ. 양자화

1. 학습 후 양자화

학습 후 양자화(Post Training Quantization)[4]는 학습된 부동 소수점 모델의 가중치 값에 양자화를 적용하는 방식이다. 학습된 모델의 가중치는 모두 다른 소수점 숫자이기 때문에 zip과 같은 간단한 압축 방식으로는 압축할 수가 없으며 각 계층에서 정규분포를 가지면서 배열, 분포된다. 각 계층의 최소/최댓값을 저장하고 스케일 계수와 연산을 통해 각각의 소수 값을 일정 범위에서 가장 가까운 실수를 나타내는 8bit 정수로 압축함으로써, 파일 크기를 줄일 수 있으며, 정수 연산으로 인한 메모리 사용량이 감소하며, 전력 소모 또한 감소한다.

가. TFLite 8비트 변환

TFLite 라이브러리에서 양자화는 위의 학습 후 양자화(Post Training Quantization) 방식을 사용하여, 이를 적용할 시 신경망의 입력부분에 양자화(Quantize) 층을 생성한다. 해당 층은 신경망으로 입력된 텐서를 INT8 형식으로 변환해 다음 층으로 주입하며, 필터의 입력 순서 또한 기존과 다르게 TFLite 모델에 맞게 변환된다.

나. ONNX런타임 8비트 변환

ONNX런타임 라이브러리 또한 TFLite와 마찬가지로 학습 후 양자화(Post Training Quantization) 방식의 양자화를 지원한다. 라이브러리의 API를 사용해 양자화를 진행한 결과, ONNX 연산자인 Mul, MatMul, Div 연산자가 추가되고 합성곱(convolution) 층 또한 변형되어, 추가된 ONNX 연산자의 연산 결과가 Cast 층을 거쳐 Mul 연산층과 하나의 결과로 계산된다. bit 관점에서 보았을 때, ONNX 모델에서 양자화 적용 후에는 MatMul 연산층이 MatMulInteger 연산층으로 바뀌어 8비트 연산 되지만 바이어스의 Mul 연산층이 추가로 생겨 Float32 연산된 후 전달된다.

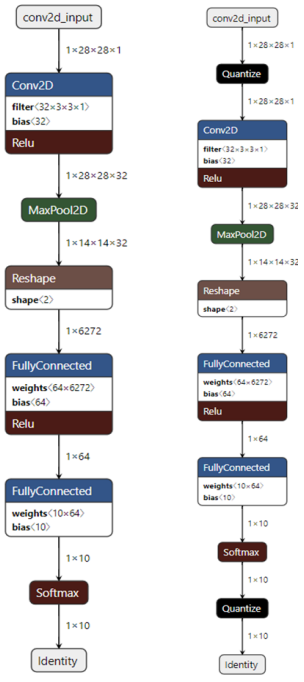

그림 2. CNN 모델의 TFLiteConverter 변환 결과 (좌 : Float32, 우 : UINT8)
Fig. 2. Result of CNN model used TFLiteConverter

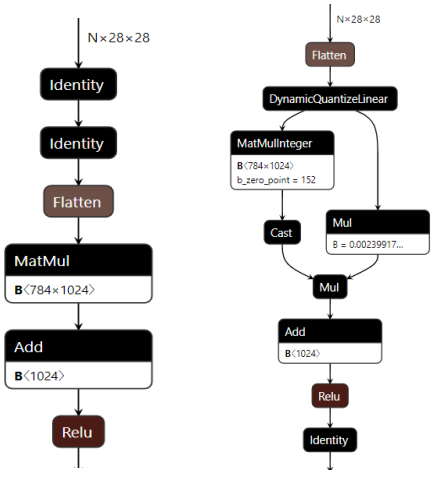

그림 3. CNN 모델의 ONNX Runtime Quantization 적용 결과
Fig. 3. Result of CNN model used ONNX Runtime Quantization

표2와 표3에서 CNN 모델과 DNN 모델 모두 ONNX 모델은 기존 모델보다 약 33%로 크기가 감소하였으며, 양자화 적용 결과 이전 모델의 약 25%로 감소하였다.

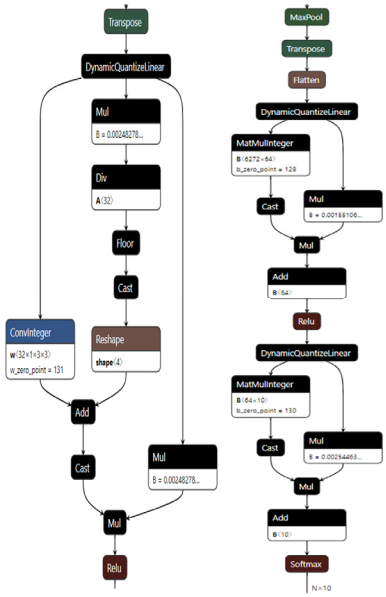

그림 4. DNN 모델의 ONNX Runtime Quantization 적용 결과

Fig. 4. Result of DNN model used ONNX Runtime Quantization

표 2. ONNX 변환 및 양자화 적용에 따른 DNN 모델 크기

Table 2. DNN model size according to ONNX transform and quantization

구분	h5	ONNX	Quantized ONNX
Size	24.00 MB	8.01 MB	2.00 MB

표 3. ONNX 변환 및 양자화 적용에 따른 CNN 모델 크기

Table 3. CNN model size according to ONNX transform and quantization

구분	h5	ONNX	Quantized ONNX
Size	9.34 MB	3.10 MB	0.78 MB

Ⅳ. 실험 및 결과

1. 추론 시간 및 정확도

Keras로 구축한 CNN[6][11] 모델과 해당 모델에 ONNX 변환과 양자화를 적용한 ONNX 모델의 정확도를 비교하였고 양자화가 적용된 TFLite 모델과 ONNX 모델의 추론 시간을 비교하였다. 또한. 합성곱(convolution) 층과 전결합(fully connected) 층의 파라미터 비중에 따른 ONNX 변환 및 양자화 효과를 비교, 분석하였다. 실험에는 Keras의 MNIST 데이터를 사용하였으며 모든 추론 시간은 1,000장을 기준으로 하였다. h5 모델은 Keras 라이브러리의 evaluate 평가 함수를 사용하였고 ONNX 모델은 ONNX런타임 라이브러리의 run 함수를 사용하여 추론 및 정확도 측정을 진행하였다.

실험 결과 양자화를 적용하여도 큰 정확도 감소가 없음을 보였으며, TFLite 라이브러리보다 ONNX 라이브러리의 양자화를 적용한 모델의 추론 속도가 더 빠른 것으로 나타났다. 한편 파라미터 비중에 따른 추론 시간 변화량 측정 결과, 네트워크가 전결합(fully conntected) 층으로 구성될 때 양자화 효과가 가장 좋았으며, 합성곱(convolution) 층이 추가되자 반대로 추론 시간이 증가하는 결과를 보였다. 이는 현재 ONNX 라이브러리의 합성곱(convolution) 층 최적화 성능이 전결합(fully connected) 층 최적화보다 뒤떨어지나, 전결합 층 최적화 수준으로 미루어 봤을 때 이후 개선이 이루어진다면 합성곱(convolution) 층의 연산시간도 대폭 감소할 수 있을 것으로 예상된다.

표 4. 양자화 전 h5 모델과 ONNX 양자화가 적용된 모델의 추론 성능

Table 4. Inference performance of h5 models before quantization and models after ONNX quantization is applied

회차	h5	ONNX INT8
	정확도	정확도
1	65.6%	65.5%
2	63.9%	63.3%
3	64.5%	63.9%
4	65.2%	65.1%
5	66.1%	66.1%

표 5. TFLite INT8 모델과 ONNX INT8 모델의 추론 시간

Table 5. Inference time of TFLite INT8 model and ONNX INT8 model

회차	TFLite INT8	ONNX INT8
1	1.904	0.125
2	2.006	0.126
3	1.975	0.124
4	2.028	0.124
5	1.935	0.136

추론시간 단위 : 초

표 6. h5모델, ONNX모델 과 ONNX INT 8 모델의 추론시간
Table 6. Inference time of model h5, ONNX and ONNX INT8 by dense rate

Dense rate	h5	ONNX	Quantize ONNX
100	22.55	0.12	0.04
89	24.17	0.49	0.84
80	24.12	0.23	0.81
70	26.81	0.40	1.15

추론시간 단위 : 초

2. 모델 추론 시 메모리 할당량 비교

h5 모델과 양자화된 ONNX 모델을 통해, 두 모델 추론 시 메모리 할당량을 비교했다. python의 tracemalloc 모듈을 이용해 PID를 호출하여 프로세스의 할당된 메모리 크기를 가져와 추론 코드 전후로 메모리 변화량을 측정했다. CPU의 연산 지원을 받는 ONNX런타임 추론과 실제 임베디드 환경을 고려해 h5 모델과 onnx 모델 모두 CPU 추론을 진행하였다. 모델은 CNN 모델과 DNN 모델을 활용했다. 데이터셋은 각각 CIFAR-10과 MNIST를 사용했다. 이미지는 1,000장 추론으로 평균 1장 추론 시 사용한 메모리 크기이다.

실험 결과 양자화 전후로 비교했을 때, 표 4와 같게 두 모델 모두 높은 정확도를 유지하였다. 게다가 표7,표8처럼 양자화 결과 CNN 모델의 메모리 사용량은 약 12.5%로 감소하였고, DNN 모델의 메모리 사용량은 약 10%로 감소하였다.

표 7. CNN h5 모델과 ONNX INT 8 모델의 메모리 할당량
Table 7. Memory useage of CNN h5 and ONNX model

회차	h5	ONNX INT8
1	8.7578	1.1133
2	8.5195	1.1328
3	8.4766	1.1094
4	8.3125	1.1172
5	8.9102	1.1484
평균	8.60	1.120

메모리 단위 : KB

V. 결론 및 향후 연구

본 연구에서는 ONNX 모델과 양자화를 통해 모델의 경량화 및 변환된 신경망의 구조를 파악하였으며, 타 경량화 라이브러리와 양자화 성능을 비교하였다. 그 결과 ONNX로 변환된 신경망은 기존 프레임워크의 층이 ONNX 고유의 층으로 변환되었으며, 모델의 크기 또한 확연히 줄어들었다. ONNX런타임 라이브러리의 양자화를 적용한 후 정확도는 1% 내외의 변동이 있었으며 추론 시간은 기존보다 약 30% 감소하였다. 또한, TFLite 라이브러리에 비해 좋은 성능을 보였다. 메모리 할당량 역시 기존의 10%-12.5%로 감소하였다. 합성곱(convolution) 층이 포함된 모델의 경우, 양자화를 적용하지 않은 ONNX 모델의 추론 시간이 더 짧았다는 사실이 개선점으로 남았다.

표 8. DNN h5 모델과 ONNX INT 8 모델의 메모리 할당량
Table 8. Memory useage of DNN h5 and ONNX model

회차	h5	ONNX INT8
1	3.9805	0.42578
2	4.0313	0.4063
3	3.8125	0.3984
4	3.9453	0.4219
5	4.5586	0.43360
평균	4.070	0.4200

메모리 단위 : KB

본 실험을 통해 인공지능 모델을 임베디드 시스템에 적용할 가능성을 확인할 수 있었다. 이후 양자화뿐만 아니라 가지치기[6]와 지식 증류 등 다양한 방법의 경량화와 최적화가 연구되고 ONNX 공유 플랫폼을 사용해 실제 환경에 적용한다면 더욱더 나은 성능과 효율을 보이는 인공지능 모델을 사용할 수 있을 것으로 예상된다.

References

[1] M. Habib ur Rehman, S. L. Chee, T. Y. Wah, A. Iqbal and P. P. Jayaraman, "Opportunistic Computation Offloading in Mobile Edge Cloud Computing Environments," 2016 17th IEEE International Conference on Mobile Data Management (MDM), Porto, 2016, pp. 208-213, DOI: https://doi.org/10.1109/MDM.2016.40.

[2] Park, Jong-Cheon, and Lee, Keun-Wang. "Mobile Phone Camera Based Scene Text Detection Using Edge and Color Quantization." Journal of the Korea Academia-Industrial cooperation Society v.11 no.3 (March 31, 2010): 847-52. DOI: 10.5762/KAIS.2010.11.3.847.

[3] S. Park and J. Heo, "Conversion Tools of Spiking Deep Neural Network based on ONNX," The journal of the institute of internet, broadcasting and

communication, vol. 20, no. 2, pp. 165-170, Apr. 2020. DOI: https://doi.org/10.7236/JIIBC.2020.20.2.165

[4] Wu, Di, et al. "EasyQuant: Post-training Quantization via Scale Optimization." Post training quantization arXiv preprint arXiv:2006.16669 (2020). DOI: arxiv-2006.16669

[5] H.-P. Kwon and J.-C. Ha, "Power Analysis Attack of Block Cipher AES Based on Convolutional Neural Network,"Journal of the Korea Academia-Industrial cooperation Society, vol. 21, no. 5, pp. 14-21, May 2020. DOI: http://dx.doi.org/10.5762/KAIS.2020.21.5.14

[6] Paupamah, Kimessha, Steven James, and Richard Klein. "Quantisation and Pruning for Neural Network Compression and Regularisation." Pruning 2020 International SAUPEC/RobMech/PRASA Conference. IEEE, 2020. DOI:10.1109/SAUPEC/RobMech/PRASA48453.2020.9041096

[7] Louis, Marcia S., Zahra Azad, Leila Delshadtehrani, S. Gupta, Pete Warden, V. Reddi and A. Joshi. "Towards Deep Learning using TensorFlow Lite on RISC-V." (2019). DOI: 10.13140/RG.2.2.30400.89606

[8] Verhelst, M. and Bert Moons. "Embedded Deep Neural Network Processing: Algorithmic and Processor Techniques Bring Deep Learning to IoT and Edge Devices." IEEE Solid-State Circuits Magazine 9 (2017): 55-65. DOI: 10.1109/MSSC.2017.2745818

[9] Lin, Wei-Fen et al. "ONNC: A Compilation Framework Connecting ONNX to Proprietary Deep Learning Accelerators." 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS) (2019): 214-218. DOI:10.1109/AICAS.2019.8771510

[10] Ashiquzzman Akm, Dong Su Lee, Sang Woo Kim, Lin Van Ma, Um-Tae Won, Jin Sul Kim "Node Pruning for Improved Neural Network Design" Korean Institute of Information Technology, 87-89, November. 2018.

[11] Bongkyu Lee. (2020). "A Study on the Analysis of Structural Textures using CNN (Convolution Neural Network)". The Journal of the Institute of Internet, Broadcasting and Communication, 20(4), 201-205. DOI : 10.7236/JIIBC.2020.20.4.201

저 자 소 개

장 두 혁(학생회원)

- 2020년 : 한성대학교 컴퓨터공학부졸업(학사)
- 2020년 ~ 현재 : 한성대학교 컴터공학과(석사)
- 관심분야 : 임베디드 시스템, 기계 학습, 경량화, 운영체제

이 정 수(학생회원)

- 2017년 ~ 현재 : 한성대학교 컴퓨터공학부(학사)
- 관심분야 : 기계학습, 네트워크 경량화, 컴퓨터 비전

허 준 영(정회원)

- 1998년 : 서울대학교 컴퓨터공학과졸업
- 2009년 : 서울대학교 컴퓨터 공학과 졸업(박사)
- 2009년 ~ 현재 : 한성대학교 컴퓨터공학부 교수
- 관심분야 : 운영체제, 무선 센서 네트워크, 임베디드 시스템, 기계 학습

※ 본 연구는 국토교통과학기술진흥원 하천조사 및 모니터링 특화 드론 플랫폼 기반 하천관리 기술개발(20DPIW-C153746-02)의 지원을 받아 수행되었습니다.