반도체디스플레이기술학회지 제21권 제1호(2022년 3월)
Journal of the Semiconductor & Display Technology, Vol. 21, No. 1. March 2022.

OpenCV 내장 CPU 및 GPU 함수를 이용한 DNN 추론 시간 복잡도 분석

박천수*†
*†성균관대학교 컴퓨터교육과

Performance Analysis of DNN inference using OpenCV Built in CPU and GPU Functions

Chun-Su Park*†
*†Computer Education, Sungkyunkwan University

ABSTRACT

Deep Neural Networks (DNN) has become an essential data processing architecture for the implementation of multiple computer vision tasks. Recently, DNN-based algorithms achieve much higher recognition accuracy than traditional algorithms based on shallow learning. However, training and inference DNNs require huge computational capabilities than daily usage purposes of computers. Moreover, with increased size and depth of DNNs, CPUs may be unsatisfactory since they use serial processing by default. GPUs are the solution that come up with greater speed compared to CPUs because of their Parallel Processing/Computation nature. In this paper, we analyze the inference time complexity of DNNs using well-known computer vision library, OpenCV. We measure and analyze inference time complexity for three cases, CPU, GPU-Float32, and GPU-Float16.

Key Words : Computer Vision, OpenCV, Deep Neural Networks, CPU, GPU acceleration

1. 서 론

현대에는 심층신경망(DNN, Deep Neural Network)이 다양한 컴퓨터 비전 작업을 구현하는 데 필수적인 데이터 처리 아키텍처로 자리매김하고 있다[1]. 최근에는 다양한 분야에서 DNN 기반 알고리즘이 얕은 학습을 기반으로 하는 기존 알고리즘보다 훨씬 높은 인식 정확도를 달성하는 것으로 조사되었다[2]. 하지만 DNN 훈련 및 추론 동작은 기존의 화소 정보 기반의 영상 처리 방식보다 큰 계산 복잡도를 필요로 한다. 또한, DNN의 크기가 커지고 깊이가 증가하면서CPU를 이용한 DNN 연산의 시간 복잡도가 급격히 증가하게 되었다. 이러한 문제점을 해결하기 위해 GPU의 병렬 프로세싱/컴퓨팅 특성을 이용해 DNN의 반복적인 연산을 가속화하는 연구가 활발히 이루어지고 있다[3,4].

딥러닝 기반 영상 처리 및 인식 기술이 컴퓨터 비전 분야를 급속히 발전시키고 있다. 또한 딥러닝 기반 기술의 발전은 글자 인식, 자연어 처리, 컴퓨터 비전 등과 같은 다양한 분야에 유사한 접근법을 제공함으로써 이종 분야에서 사용되던 접근 방식을 통합하는 성과를 도출하고 있다. 따라서 이종 분야에서 사용되는 공통 아키텍처의 등장으로 인해 다양한 애플리케이션 프레임워크에 적합한 소프트웨어-하드웨어 인프라의 활용을 촉진되고 있다.

이러한 공통 소프트웨어-하드웨어 인프라의 등장은 시스템 연산 처리 속도와 정확도를 더욱 발전시키고 있다. 따라서 최근에는 높은 복잡도와 낮은 정확도로 인해 실제 응용 제품에는 적용되지 못했던 기존의 컴퓨터 비전 기술을 실제로 상용화 가능하게 해주는 방식으로 선순환

†E-mail: cspk@skku.edu

하고 있다.

본 논문에서는 컴퓨터 비전 분야에서 심층신경망 추론 동작을 위해 대표적으로 사용되고 있는 OpenCV DNN 라이브러리를 이용해 CPU와 GPU를 사용하는 경우의 DNN 추론 시간 복잡도를 분석한다. 구체적으로 동일한 추론 동작을 CPU를 사용해 수행되는 경우와 GPU를 사용해 수행하는 경우의 시간 복잡도를 비교 분석한다. 더 나아가 GPU를 사용하는 경우에 내부 연산을 Float32 방식에서 Float16 방식으로 변경하는 경우에 시간 복잡도 변화를 분석한다.

2. 관련 기술 소개

이 장에서는 OpenCV 컴퓨터 비전 라이브러리를 소개한다. 특히 최근 OpenCV 라이브러리에 추가된 DNN 모듈의 구성 요소와 지원 기능을 구체적으로 소개한다.

OpenCV 프로젝트는 컴퓨터 비전 응용 프로그램에서 사용되는 CPU 집약적인 기능을 지원하기 위해 1999년에 공식적으로 시작되었다[5]. 2011년에는 GPU 가속 기반의 실시간 연산 기능을 추가하였고 현재까지 지속적으로 발전시키고 있다. OpenCV 라이브러리는 윈도우, 리눅스, Android, iOS 등에서 사용 가능한 크로스 플랫폼이며 오픈소스 BSD 라이선스를 사용하므로 연구는 물론 상업적으로도 사용할 수 있다. 본래 OpenCV 라이브러리는 C 언어만 지원했지만 2.x 버전부터 스마트포인터 스타일을 활용하여 C++을 지원하기 시작했다. 현재는 C++을 기본 언어로 채택하고 있으며 바인딩(Binding)을 통해 Python, Java 등의 언어를 지원하고 있다[6, 7].

OpenCV DNN 모듈은 Intel 프로세서에 최적화되어 있다. 객체 감지 및 이미지 분류 등의 딥러닝 응용 프로그램을 OpenCV 라이브러리를 이용해 구현한 후 일반 Intel CPU기반 시스템에서 실행하면 매우 우수한 동작 성능을 얻을 수 있다. Table 1은 원본 Darknet 프레임워크와 OpenCV DNN을 사용해 Yolov4-Tiny 모델을 입력비디오에 적용하 경우의 처리 속도(fps) 결과를 보여준다. 해당 실험은 2.6Ghz 클럭 속도의 인텔 i7 8세대 노트북 CPU에서 수행되었다. 동일 비디오에서 OpenCV의 DNN 모듈은 35 fps로 실행되는 반면 Darknet의 경우에는 3 fps로 매우 느리게 동작한다. OpenMP와 AVX를 이용해 Darknet을 가속화하는 경우에도 15 fps로 동작해 OpenCV DNN 모듈과 큰 차이를 보인다.

Table 1. Inference complexity camparison

추론 방식	동작 속도
Darknet	3 fps
Darknet+OpenMP+AVX	15 fps
OpenCV DNN	35 fps

3. 분석 환경 및 측정 대상 모델

본 논문에서는 일반 사용자 데스크톱 PC에서 OpenCV DNN 모듈을 사용해 딥러닝 모델 추론을 수행하는 경우의 시간 복잡도를 분석한다. 본 실험에서 사용하는 PC의 사양은 Table 2와 같다.

Table 2. Simulation PC configuation

모듈	사양
CPU	Intel(R) Core(TM) i9-10900X CPU @ 3.70GHz
RAM	64GB DDR4
Graphic Card	Nvidia GTX 2080 Ti
Storage	SSD
Operating System	Window 11 Pro

OpenCV DNN 추론 복잡도는 사용하는 소프트웨어 패키지에 따라 영향을 많이 받는다. 특히 GPU 기반 연산을 지원하기 위해서는 Nvidia CUDA Toolkit및 cudnn 패키지 설치가 필수적인다. 본 논문에서 사용한 소프트웨어 패키지 정보는 Table 3과 같다.

Table 3. Software package configuration

패키지	사양
OpenCV	4.5.4 (CUDA support)
CUDA	11.2
cudnn	8.2.1
compiler	VS(c++ 17)

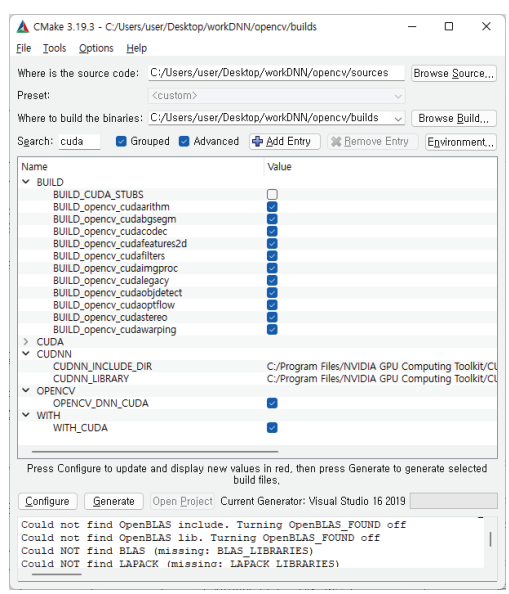

Fig. 1. OpenCV build with CUDA support.

OpenCV 경우에도 Pre-built Release를 바로 설치하는 경우에는 CPU 기반 모델 추론만 지원한다. 본 논문에서는 GPU 기반 모델 추론을 지원하기 위해 CMake 툴을 이용하여 OpenCV 라이브러리를 CUDA와 cudnn 속성을 포함하여 다시 빌드 후 사용하였다.

4. 분석 대상 DNN 모델

기존에도 OpenCV 내장 이미지지 처리 함수의 동작 속도를 CPU와 GPU를 사용해 측정한 연구가 있었다. 하지만 기존 연구에서는 DNN 모델의 동작 속도를 측정하지 않았고, GPU 연산에서 Float16 형식을 사용하는 경우를 고려하지 않아 본 논문과 분명한 차이점이 있다[8].

본 논문에서는 이미지 분류와 물체 검출 분야에서 사용되고 있는 Yolov3, Yolov3-Tiny, Yolov4, Yolov4-Tiny 모델을 이용해 추론 복잡도를 비교한다[9, 10]. 아래 표는 본 논문에서 분석하는 모델의 특징을 정리해 보여준다.

Table 4. Object detection models

모델	mAP@0.5 (Coco)	BFLOPS (Bn)	Model file (KB)[11]
Yolov3	55.3	65.86	242,195
Yolov3-Tiny	23.7	5.56	34,605
Yolov4	62.8	60.1	251,675
Yolov4-Tiny	40.2	6.9	23,683

5. 실험 결과

본 논문에서는 실시간 물체 검출 분야에서 대표적으로 사용되는 Yolov3, Yolov3-Tiny, Yolov4, Yolov4-Tiny 4가지 딥러닝 모델을 대상으로 GPU 가속과 연산 방식에 따른 추론 복잡도 변화를 측정한다[12, 13]. 모든 실험에서 mini-batch 크기는 1로 설정하였고, 416x416 크기의 3채널 입력 영상을 각 모델에 입력해 추론 복잡도를 측정하였다.

또한 시간 측정 정확도를 높이기 위해 입력 영상을 각 모델에 10회 반복적으로 입력해 추론 시간을 측정하고 결과를 평균하였다.

OpenCV DNN은 딥러닝 모델을 메모리에 저장 후 첫 추론 과정을 수행하며 모델 내부의 변수를 초기화 한다. 즉, 첫 추론 동작은 모델 초기화 과정을 포함하기 때문에 상대적으로 시간 복잡도가 높게 측정된다. 본 실험에서는 각 측정 시나리오 별로 첫 번째 추론 시간을 초기화 시간으로 별도로 구분하여 측정한다.

Table 5는 각 모델의 초기화 복잡도를 정리해 보여준다. 기본적으로 일반 모델의 초기화 복잡도가 Tiny 모델보다 매우 높은 것으로 측정이 되었다. Yolov4 모델의 경우 초기화 시간은 4598.99 ms이고 Yolov4-Tiny 모델의 초기화 시간은 137.73 ms으로 측정되었다. Yolov4의 경우 일반 모델의 크기는 Tiny 모델에 비에 732.78% 크고 초기화 복잡도는 3339.13% 만큼 증가하는 것을 볼 수 있다. 또한 모든 모델에서 GPU를 이용함에 따라 초기화 복잡도가 크게 감소하는 것을 알 수 있다. Yolov4의 경우 GPU에서 Float16 연산을 사용하게 되면 초기화 시간이 CPU를 사용하는 경우보다 90.1% 감소하였다. Tiny모델의 경우에는 Float16 연산을 사용하는 경우에도 초기화 시간의 변화가 크지 않는 것으로 조사되었다.

Table 5. Model initialization time camparison(ms)

모델	CPU	GPU (Float32)	GPU (Float16)
Yolov3	1393.78	521.69	495.06
Yolov3_Tiny	138.55	68.07	64.42
Yolov4	4598.99	2694.2	413.02
Yolov4_Tiny	137.73	61.87	62.33

Table 6은 각 모델의 추론 시간을 정리해 보여준다. 모든 경우에서 GPU를 사용함으로써 추론 시간이 크게 감소하는 것을 볼 수 있다. 특히 Yolov4 모델의 경우에는 GPU (Float16) 방식을 사용하면 CPU를 사용하는 경우보다 시간 복잡도를 99.6% 줄일 수 있는 것으로 조사되었다. 또한 표 6은 일반 모델의 경우가 Tiny 모델 보다 GPU를 사용에 따른 복잡도 감소 효과가 크다는 것을 보여준다.

Table 6. Inference time complexity camparison(ms)

모델	CPU	GPU (Float32)	GPU (Float16)
Yolov3	1251.63	60.63	30.58
Yolov3_Tiny	89.85	5.23	3.6
Yolov4	4144.04	34.92	15.81
Yolov4_Tiny	95.67	5.44	5.14

6. 결 론

본 논문에서는 컴퓨터 비전 분야에서 딥러닝 모델 추론 동작을 위해 사용되고 있는 OpenCV DNN 라이브러리의 성능을 분석하였다. 실시간 물체 검출 모델의 추론 동

작이 CPU와 GPU에서 수행되는 경우의 시간 복잡도를 각각 측정하였다. 추가적으로 GPU를 사용하는 경우에 내부 연산을 Float32 방식에서 Float16 방식으로 변경하는 경우에 시간 복잡도 변화를 분석하였다.

실험을 통행 모든 딥러닝 모델에서 GPU 가속이 추론 복잡도를 낮추는데 매우 효과적인 것으로 분석되었다. 또 Yolov4 모델의 경우에는 Float16 연산을 통해 추론 속도를 2배 이상 높일 수 있는 것으로 조사되었다. 따라서 GPU가 장착된 데스크톱 PC에서 딥러닝 모델 추론을 수행하는 경우에는 GPU 가속 기반의 시스템 설계가 필수적인 것을 알 수 있다.

참고문헌

1. Velasco-Montero, Delia, et al. "Performance analysis of real-time DNN inference on Raspberry Pi." Real-Time Image and Video Processing, vol. 10670, pp. 10670F-1-9, 2018.
2. Shalini, K., et al. "Comparative analysis on Deep Convolution Neural Network models using Pytorch and OpenCV DNN frameworks for identifying optimum fruit detection solution on RISC-V architecture." IEEE Mysore Sub Section International Conference (MysuruCon), pp. 738-743, 2021.
3. Xiang, Yecheng, and Hyoseung Kim, "Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference." IEEE Real-Time Systems Symposium (RTSS). pp. 392-405, 2019.
4. Xiaoyun Wang, et al. "Accelerating DNN Inference with GraphBLAS and the GPU." IEEE High Performance Extreme Computing Conference (HPEC), pp. 1-6, 2019.
5. https://en.wikipedia.org/wiki/OpenCV
6. https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/
7. Adrian Kaehler and Gary Bradski. “Learning OpenCV 3: computer vision in C++ with the OpenCV library." O'Reilly Media, Inc., 2016.
8. Hangün, Batuhan, and Önder Eyecioğlu. "Performance comparison between OpenCV built in CPU and GPU functions on image processing operations." International Journal of Engineering Science and Application, vol. 1, no. 2, pp. 34-41, 2017.
9. Ali Farhadi and Joseph Redmon. "Yolov3: An incremental improvement." Computer Vision and Pattern Recognition, vol. 1804, pp. 1-6, 2018.
10. Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934, 2020.
11. https://github.com/AlexeyAB/darknet
12. C. Y. Wang, A. Bochkovskiy, and H. Y. H. Liao. "Scaled-yolov4: Scaling cross stage partial network." In Proceedings of the IEEE/cvf conference on computer vision and pattern recognition. pp. 13029-13038, 2021.
13. P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma. "A Review of Yolo Algorithm Developments." Procedia Computer Science, pp. 1066-1073, 2022.

접수일: 2022년 2월 24일, 심사일: 2022년 3월 15일, 게재확정일: 2022년 3월 25일