반도체디스플레이기술학회지 제21권 제3호(2022년 9월)
Journal of the Semiconductor & Display Technology, Vol. 21, No. 3. September 2022.

다양한 컴퓨팅 환경에서 YOLOv7 모델의 추론 시간 복잡도 분석

박천수*†
*†성균관대학교 컴퓨터교육과

YOLOv7 Model Inference Time Complexity Analysis in Different Computing Environments

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

ABSTRACT

Object detection technology is one of the main research topics in the field of computer vision and has established itself as an essential base technology for implementing various vision systems. Recent DNN (Deep Neural Networks)-based algorithms achieve much higher recognition accuracy than traditional algorithms. However, it is well-known that the DNN model inference operation requires a relatively high computational power. In this paper, we analyze the inference time complexity of the state-of-the-art object detection architecture Yolov7 in various environments. Specifically, we compare and analyze the time complexity of four types of the Yolov7 model, YOLOv7-tiny, YOLOv7, YOLOv7-X, and YOLOv7-E6 when performing inference operations using CPU and GPU. Furthermore, we analyze the time complexity variation when inferring the same models using the Pytorch framework and the Onnxruntime engine.

Key Words : YOLOv7, Deep neural networks, Pytorch, Onnxruntime, GPU acceleration

1. 서 론

물체 검출 기술은 컴퓨터 비전 분야의 주요 연구 주제 중 하나로 다양한 비전 시스템을 구현하는데 필수 기반 기술로 자리매김하고 있다. 현대에는 심층신경망(DNN, Deep Neural Network)을 이용한 물체 검출 기술이 활발히 연구되고 있으며 보행자와 같이 형태가 변하는 물체도 정확히 탐지하는 수준으로 성능이 개선되었다[1, 2]. 따라서 보안 카메라[3, 4], 자율 주행[5], 의료 이미지 분석[6] 시스템 등 실시간으로 입력되는 영상에서 대상 물체를 검출하는 기능이 필요한 상용 서비스에 활발히 도입되고 있다.

최근의 심층신경망 기술과 하드웨어 컴퓨팅 능력의 발전은 물체 검출 기술의 급격한 발전을 이끌었다. 최첨단 물체 검출 기술은 깊은 CNN(Convolutional Neural Network) 레이어 층을 사용하는 경우에도 최소한의 데이터를 이용해 더 복잡한 기능을 학습할 수 있게 발전하였다.

물체 검출 기술은 일반적으로 2가지 범주로 구분이 가능하다[7]. 영역 제안(region proposal)에 기반한 2-stage 기술과 동시 회귀-분류에 기반한 1-stage 기술이다.

2-stage 검출 기술은 우선 전체 영상을 검사해 물체가 존재할 가능성이 높은 영역을 찾고 해당 영역에 전통적인 물체 검출 파이프라인을 적용한다. 즉, 첫 단계에서는 후보 경계 영역 ROI(regions of interest)를 검출하고, 두 번째 단계에서는 각 ROI별로 특징을 추출하여 분류 및 경계 상자 회귀 작업을 수행한다. 널리 사용되는 2-stage 검출기에는 R-CNN[8], Fast R-CNN[9], R-FCN[10], Fast R-CNN[11],

†E-mail: cspk@skku.edu

Mask R-CNN[12] 등이 있다.

이와는 다르게 1-stage 물체 검출 기술은 영역 검출과 회기/분류 동작을 통합한 프레임워크를 채택하여 영역 검출 단계를 생략하였다. 따라서 1-stage 물체 검출 기술은 이미지 픽셀에서 물체의 좌표, 레이블, 클래스 확률을 바로 매핑한다. 가장 잘 알려진 1-stage 검출기는 SSD(Single Shot MultiBox Detector) 계열 기술과 YOLO(You Only Look Once) 계열 기술 등이 있다[13, 14].

본 논문에서는 최신 1-stage 물체 검출 기술인 Yolov7의 추론 시간 복잡도를 다양한 환경에서 분석한다. 구체적으로 4가지 유형의 Yolov7 모델(YOLOv7-tiny, YOLOv7, YOLOv7-X, YOLOv7-E6)의 추론 동작을 CPU를 사용해 수행하는 경우와 GPU를 사용해 수행하는 경우의 시간 복잡도를 비교 분석한다. 더 나아가 동일한 모델을 Pytorch 프레임워크와 Onnxruntime 엔진을 이용해 추론하는 경우의 시간 복잡도 변화를 분석한다.

2. YOLO기술 소개

YOLO는 2015년에 처음 소개가 된 후 많은 컴퓨터 비전 연구자들로부터 관심을 받아왔다. YOLO 기술은 1-stage 방식의 통합 프레임워크를 채택해 입력 영상에서 대상 물체의 영역, 레이블, 클래스 확률을 바로 매핑한다. YOLOv1의 주요 아이디어는 고정된 크기의 그리드 셀을 이미지에 적용하는 것이다. 물체의 중심이 속하는 그리드 셀이 해당 물체를 탐지하는 역할을 담당하고 나머지 그리드 셀은 해당 영역에 물체가 존재하더라도 물체의 중심 좌표를 포함하지 않는다면 검출 역할을 수행하지 않는다.

배치 정규화는 딥러닝 모델 설계에서 가장 인기 있는 정규화 방법 중 하나로 입력 계층의 분포를 안정화시켜 심층 신경망의 더 빠르고 안정적인 훈련을 가능하게 한다. YOLOv2에서는 배치 정규화 기법을 도입하여 검출 정확도를 높였다[15]. 또한 YOLOv2는 Darket-19 백본(backbone)과 기준 상자(Anchor box)를 이용한 영역 검출 방식을 채택했다.

YOLOv3는 Darknet-53 백본과 잔류 네트워크 (ResNet) 구조를 도입한 새로운 아키텍처를 도입했다[16](Fig. 1). 또한 멀티스케일(multi-scale) 검출 방식을 채택해 3개의 백본 레이어에 예측 레이어를 각각 덧붙여 다양의 크기의 물체를 효과적으로 검출하는 것을 가능하게 했다.

2020년 4월에는 Alexey Bochkovsky가 YOLOv4를 발표했다[17]. YOLOv4는 SPP(Spatial Pyramid Pooling) 블럭과 CSPdarknet53 백본을 채택하고 데이터 증강(Data Augmentation) 등의 다양한 Bag of Freebies 기법을 이용해 최종 모델의 검출 성능을 높였다. YOLOv4 구조는 후에 PP-YOLO, Scaled-YOLOv4, YOLOX, YOLOR 등의 구조로 더욱 개선되었다[19].

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 1. Darkent-53 backbone[16].

YOLOv5는 2020년 6월에 Glenn Jocher가 발표했고 기존의 YOLO 시리즈와는 다른 백본 구조를 채택했다[18]. 성능면에서 YOLOv4와 유사하지만 사용자 편의성이 높은 구현 방식을 채택해 많은 분야에서 사용되고 있다. 구체적으로 YOLOv5에는 K-mean Anchors, Generic Learning 등의 기술이 채택되었다.

2022년에는 YOLOv6와 YOLOv7가 비슷한 시기에 발표되었다. YOLOv6는 정식 YOLO 시리즈는 아니고 하드웨어 친화적인 설계와 고성능을 갖춘 산업 애플리케이션을 대상으로 개발되었다[20]. YOLOv7는 5 ~ 160 fps 범위의 추론 속도를 가지면 현재까지 알려진 실시간 물체 검출 모델 중 가장 높은 정확도인 56.8% AP를 가지는 것으로 발표되었다[21]. 구체적으로 새로운 bag-of-freebies 방법과 Extended-ELAN (E-ELAN) 구조를 도입해 SOTA(state-of-the-art) 성능을 달성했다(Fig. 2). 본 논문에서는 YOLOv7모델의 추론 복잡도를 다양한 컴퓨팅 환경에서 측정하고 결과를 분석한다.

3. 분석 환경 및 측정 대상 모델

본 논문에서는 일반 사용자 데스크톱 PC에서 입력영상 크기, 프로세싱 유닛, 소프트웨어 엔진을 바꿔가며 최신

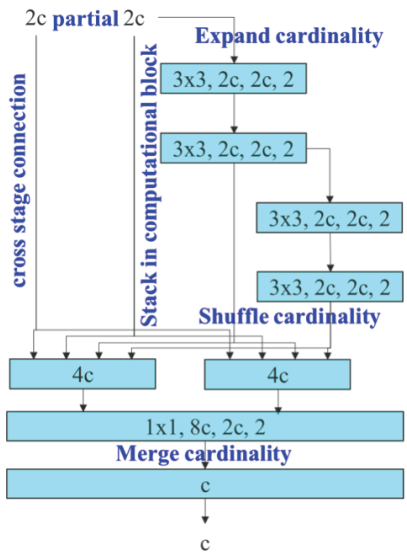

Fig. 2. E-ELAN 구조[21].

Table 1. 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

Table 2. Software package configuration

패키지	사양
Pytorch	1.12(CUDA support)
Onnxruntime-gpu	1.12
CUDA	11.6
cudnn	8.4.1

YOLOv7 모델의 추론 시간 복잡도를 분석한다. 본 논문에서 사용하는 시뮬레이션 PC의 사양은 Table 1과 같다.

딥러닝 모델 추론 복잡도는 사용하는 소프트웨어 패키지의 영향을 많이 받는다. 특히 GPU 기반 연산을 지원하기 위해서는 Nvidia CUDA Toolkit및 cudnn 패키지 설치가 필수적인다. 본 논문에서 사용한 소프트웨어 패키지 정보는 Table 2와 같다.

본 논문에서는 최근 소개된 YOLOv7 모델 4가지 유형의 추론 복잡도를 분석한다. 구체적으로 YOLOv7-tiny, YOLOv7, YOLOv7-X, YOLOv7-E6 모델을 분석한다. Table 3은 본 논문에서 분석하는 4가지 모델의 특징을 정리해 보여준다. 참고로 모델의 AP 성능은 YOLOv7-tiny의 경우는 416x416,

Table 3. Object detection models

모델	AP @(0.5:0.95)	#Param (M)	Model file (KB)[11]
Yolov7-tiny	35.2	6.2	12,344
Yolov7 (Basic)	51.2	36.9	73,816
Yolov7-X	52.9	71.3	139,746
Yolov7-E6	55.9	97.2	190,470

YOLOv7의 경우는 640x640, YOLOv7-X의 경우는 640x640, YOLOv7-E6의 경우는 1280x1280 크기의 입력 영상을 기준으로 측정되었다. 또 모델 파일 크기는 모델 파라미터를 FP16 방식으로 저장하고 측정하였다.

4. 실험 결과

본 논문에서는 최신 실시간 물체 검출 모델 YOLOv7의 4가지 유형 YOLOv7-tiny, YOLOv7, YOLOv7-X, YOLOv7-E6 모델을 대상으로 추론 방식, 프로세싱 유닛, 입력영상 크기를 바꿔가며 추론 시간 복잡도 변화를 측정한다. 모든 실험에서 mini-batch 크기는 1로 설정하였고, 3채널 RGB 영상을 각 모델에 입력해 추론 시간을 측정하였다. 또한 시간 측정 정확도를 높이기 위해 입력 영상을 각 모델에 100회 반복적으로 입력해 추론 시간을 측정하고 결과를 평균하였다.

일반적으로 딥러닝 모델 추론 동작은 주 메모리에 해당 모델을 복사한 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[22].

Table 4는 각 모델의 초기화 복잡도를 정리해 보여준다. 기본적으로 모델의 크기가 증가할수록 초기화 복잡도가 증가하는 것을 볼 수 있다. YOLOv7-tiny 모델을 Pytorch 프레임워크를 이용해 CPU에서 320x320 영상을 입력으로 추론을 수행할 경우 52.48ms가 소요되었다. 같은 환경에서 일반 YOLOv7의 경우 초기화 시간은 239.83ms으로 추론 시간이 급격히 증가하는 것을 볼 수 있다. 다만 YOLOv7-E6 모델의 경우 Pytorch 프레임워크를 이용하는 경우에는 크기가 상대적으로 작은 YOLOv7-X 모델 보다 초기화 시간이 오래 걸리는 것으로 조사되었다. 이와는 다르게 Onnxruntime 엔진을 이용하는 경우에는 모델의 크기가 클수록 초기화 시간이 지속적으로 증가하는 것을 볼 수 있다.

모든 조건에서 입력 영상의 크기가 커질수록 초기화

Table 4. Model initialization time camparison(ms)

Inference Method	Processing Unit	Input Size	Models			
			Yolov7-tiny	Yolov7	Yolov7-X	Yolov7-E6
Pytorch (*.pt)	CPU	320 x 320	52.48	239.83	326.60	294.39
		640 x 640	154.12	734.06	1010.42	823.05
	GPU	320 x 320	17.96	34.29	58.54	77.06
		640 x 640	19.91	38.46	82.48	124.96
Onnxruntime (*.onnx)	CPU	320 x 320	13.10	59.34	87.39	123.11
		640 x 640	48.54	194.46	515.94	403.97
	GPU	320 x 320	247.47	374.24	479.87	719.10
		640 x 640	256.22	447.37	596.63	823.00

Table 5. Inference time complexity camparison(ms)

Inference Method	Processing Unit	Input Size	Models			
			Yolov7-tiny	Yolov7	Yolov7-X	Yolov7-E6
Pytorch (*.pt)	CPU	320 x 320	49.80	236.34	322.74	292.32
		640 x 640	154.74	727.83	1010.73	820.49
	GPU	320 x 320	15.15	25.62	45.06	49.31
		640 x 640	15.61	26.06	46.10	49.77
Onnxruntime (*.onnx)	CPU	320 x 320	11.18	42.72	69.78	109.54
		640 x 640	53.38	176.72	509.92	380.02
	GPU	320 x 320	11.34	15.76	19.02	24.70
		640 x 640	17.25	27.94	38.03	36.73

시간이 길어지는 현상을 관찰할 수 있다. 다만 GPU를 이용하는 경우에는 입력영상이 커지는 경우에도 초기화 시간이 상대적으로 작게 증가하는 경향을 보인다. Pytorch 프레임워크 기반 추론과 Onnxruntime을 이용한 추론 동작을 비교해보면 CPU의 경우에는 Pytorch 방식이, GPU의 경우에는 Onnxruntime 방식이 초기화 시간이 오래 걸리는 것으로 조사되었다.

Table 5는 각 모델의 초기화 시간을 제외한 평균 추론 시간을 정리해 보여준다. Table 4와 Table 5를 비교해 보면 대부분의 경우에 초기화 시간보다 일반 추론 시간이 더 짧을 것을 볼 수 있다. 또, Onnxruntime 방식의 경우 초기화 시간과 추론 시간의 차이가 더욱 큰 것을 볼 수 있다.

대부분의 경우에 CPU를 사용하는 것보다 GPU를 사용함으로써 추론 시간이 감소하는 것을 볼 수 있다. 다만 Onnxruntime 방식에서는 입력 이미지의 크기가 작고 모델의 크기 작은 경우에는 CPU대신 GPU를 사용하는 것이 오히려 평균 추론 시간을 약간 증가시키는 것으로 조사되었다. 즉, 320x320 영상을 YOLOv7-tiny 모델에 입력해 추론하는 동작을 CPU를 이용해 수행하면 11.18ms가 걸렸으나, GPU를 이용하면 11.34ms가 걸리는 것으로 조사되었다.

모든 실험에서 CPU를 이용하는 경우에는 Pytorch 방식보다 Onnxruntime 방식의 추론 시간 복잡도가 더 낮은 것으로 조사되었다. GPU의 경우에는 Onnxruntime의 초기화 복잡도가 Pytorch의 복잡도가 크게 높은 것으로 조사되었다. 평균 추론 시간의 경우 일부 경우에는 Pytorch 방식의 추론 시간 복잡도가 Onnxruntime 방식보다 더 낮은 것으로 조사되었으나, 일반적으로 Onnxruntime 방식이 상대적으로 우수한 성능을 보였다. 특히 모델의 크기가 커지는 경우에는 두 추론 방식의 성능차이 커지는 것으로 조사되었다.

5. 결론

본 논문에서는 실시간 물체 검출 분야의 최신 YOLOv7 모델을 대상으로 추론 시간 복잡도 분석을 수행하였다. 실시간 물체 검출 모델의 추론 동작이 CPU와 GPU에서 수행되는 경우의 시간 복잡도를 각각 측정하였다. 추가적으로 추론 동작을 Pytorch 프레임워크와 Onnxruntime 엔진으로 수행하는 경우의 시간 복잡도 변화를 분석하였다.

실험을 통해 일반적으로 GPU 가속이 딥러닝 모델의 추론 복잡도를 낮추는데 매우 효과적인 것으로 분석되었다. 또, CPU를 이용해 YOLOv7 모델의 추론동작을 수행하는 경우에는 Pytorch 방식보다는 Onnxruntime을 이용하는 것이 효과적인 것으로 조사되었다. GPU가속을 사용해 딥

러닝 모델을 추론하는 경우에는 Pytorch 프레임워크와 Onnxruntime 엔진의 동작 성능을 비교해 너 낮은 시간 복잡도를 가지는 방식을 선택하는 과정이 필요하다. 하지만 CPU를 이용해 딥러닝 모델을 추론하는 경우에는 Onnxruntime 방식을 이용하는 것이 효과가 높은 것으로 조사되었다.

감사의 글

본 연구는 중소벤처기업부의 연구비지원(S3147433)에 의해 수행되었습니다.

참고문헌

1. Liu, Yang et al., “A survey and performance evaluation of deep learning methods for small object detection,” Expert Systems with Applications, vol. 172, no. 15, pp. 114602-1-14, 2021.
2. S. S. AbbasZaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "Velasco-Montero, Delia, et al. “A survey of modern deep learning based object detection models.” Real-Time Image and Video Processing, vol. 10670, no. 30, pp. 103514-1-9, 2022.
3. S. Jha, C. Seo, F. Yang, and G. P. Joshi, “Real time object detection and tracking system for video surveillance system.” Multimedia Tools and Applications, vol. 80, no. 3, pp. 3981-3996, 2021.
4. R. Chandrakar, R. Raja, R. Miri, U. Sinha, A. K. S. Kushwaha, and H. Raja, “Enhanced the moving object detection and object tracking for traffic surveillance using RBF-FDLNN and CBF algorithm.” Expert Systems with Applications, vol. 191, pp. 116306-1-15, 2022.
5. E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, “A survey on 3d object detection methods for autonomous driving applications.” IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 10, pp. 3782-3795, 2019
6. E. H. Nguyen, H. Yang, R. Deng, Y. Lu, Z. Zhu, J. T. Roland, and Y. Huo, “Circle Representation for Medical Object Detection.” IEEE transactions on medical imaging, vol. 41, no. 3, 746-754. 2021.
7. D. Pestana, et al. “A full featured configurable accelerator for object detection with YOLO.” IEEE Access, vol. 9, pp. 75864-75877, 2021.
8. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 580–587, Jun. 2014.
9. R. Girshick, “Fast R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), pp. 1440–1448, Dec. 2015.
10. J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object detection via region based fully convolutional networks,” in Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NIPS). Red Hook, NY, USA: Curran Associates, pp. 379–387, 2016.
11. S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
12. K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), pp. 2980–2988, Oct. 2017.
13. W. Liu, et al. “Ssd: Single shot multibox detector." European conference on computer vision. Cham, pp. 21-37, 2016.
14. P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma. “A Review of Yolo Algorithm Developments.” Procedia Computer Science, pp. 1066-1073, 2022.
15. J. Redmon, and A. Farhadi, “YOLO9000: better, faster, stronger.” In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263-7271, 2017.
16. Ali Farhadi and Joseph Redmon. “Yolov3: An incremental improvement.” Computer Vision and Pattern Recognition, vol. 1804, pp. 1-6, 2018.
17. Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. “Yolov4: Optimal speed and accuracy of object detection.” arXiv preprint arXiv:2004.10934, 2020.
18. 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.
19. W. Wu, et al. “Application of local fully Convolutional Neural Network combined with YOLO v5 algorithm in small target detection of remote sensing image.” PloS one, vol. 16, no. 10, pp. e0259283. 2021.
20. https://github.com/meituan/YOLOv6
21. C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.” arXiv preprint arXiv:2207.02696.
22. C. S. Park, “Performance Analysis of DNN inference using OpenCV Built in CPU and GPU Functions.” Journal of the Semiconductor & Display Technology, vol. 21. no. 1. pp. 75-78, 2022.

접수일: 2022년 7월 31일, 심사일: 2022년 9월 6일,
게재확정일: 2022년 9월 6일