객체 검출을 위한 트랜스포머와 공간 피라미드 풀링 기반의 YOLO 네트워크

*권오준 **정제창
한양대학교 융합전자공학과
*ojkwon1@hanyang.ac.kr, **jjeong@hanyang.ac.kr

Transformer and Spatial Pyramid Pooling based YOLO network for Object Detection

*Kwon, Oh-Jun **Jeong, Je-Chang
Department of Electronic Engineering, Hanyang University

요약

일반적으로 딥러닝 기반의 객체 검출(Object Detection)기법은 합성곱 신경망(Convolutional Neural Network, CNN)을 통해 입력된 영상의 특징(Feature)을 추출하여 이를 통해 객체 검출을 수행한다. 최근 자연어 처리 분야에서 획기적인 성능을 보인 트랜스포머(Transformer)가 영상 분류, 객체 검출과 같은 컴퓨터 비전 작업을 수행하는데 있어 경쟁력이 있음이 드러나고 있다. 본 논문에서는 YOLOv4-CSP의 CSP 블록을 개선한 one-stage 방식의 객체 검출 네트워크를 제안한다. 개선된 CSP 블록은 트랜스포머(Transformer)의 멀티 헤드 어텐션(Multi-Head Attention)과 CSP 형태의 공간 피라미드 풀링(Spatial Pyramid Pooling, SPP) 연산을 기반으로 네트워크의 Backbone과 Neck에서의 feature 학습을 돕는다. 본 실험은 MSCOCO test-dev2017 데이터 셋으로 평가하였으며 제안하는 네트워크는 YOLOv4-CSP의 경량화 모델인 YOLOv4s-mish에 대하여 평균 정밀도(Average Precision, AP)기준 2.7% 향상된 검출 정확도를 보인다.

1. 서론

Graphics Processing Unit(GPU) 성능의 발전으로 다량의 연산 처리가 가능해지면서 딥러닝 기술을 활용한 컴퓨터의 사물 인지능력은 처리 속도와 정확성 측면에서 사람을 대신할 수 있는 수준에 이르렀다. 이러한 기술의 발전에 따라 자율 주행, CCTV, 의료영상, 로봇 제어, 군사 보안장치 등 카메라가 내장된 다양한 응용 분야에 적용되고 있어 정보통신산업 전반의 핵심 기술로 주목받는 추세이다.

컴퓨터 비전 영역에서의 객체 검출 기술은 영상 속 물체가 어떤 종류인지 분류(Classification)하는 문제와 해당 물체가 어느 위치에 존재하는지를 탐색(Localization)하는 것을 목표로 한다. 이 두 가지 작업을 순차적으로 수행하는 방식을 two-stage 계열의 검출기라고 하고, 하나의 신경망으로 동시에 수행하는 방식을 one-stage 계열의 검출기라고 부른다. 일반적으로 one-stage 계열의 검출기가 two-stage 계열보다 검출 정확도는 낮지만, 빠른 검출 속도를 보인다.

최근 기계번역, 텍스트 분류, 문서 요약과 같은 자연어 처리 분야에서 획기적인 성능을 보인 Transformer [1] 가 영상 분류, 객체 검출, 영상 분할과 같은 영상 분석에도 활용되고 있다 [2]. Transformer는 자기주의(Self-attention)연산을 통해 입력 영상을 1차원 벡터 형태로 변환하여 feature 학습을 수행하므로 수용영역의 제한이 없다. 따라서 먼 위치의 화소 정보를 고려할 수 있어 광범위한 영역에 대한 feature 학습에 효과적이다. 이와 달리 CNN 기반의 feature 학습은 고정된 크기의 합성곱(Convolution) 필터를 사용하기 때문에 수용영역이 제한된다. 따라서 영상 전반의 feature 통합을 위한 다수의 convolution 계층이 필요하며 이에 따른 학습 및 추론 속도의 저하가 발생할 수 있다. 이러한 문제를 해결하고자 Scaled-YOLOv4 [3]에서는 YOLOv4 [4] 의 Backbone과 Neck에 Cross Stage Partial Network(CSPNet) [5] 를 적용한 YOLOv4-CSP 모델을 제안한다.

본 논문에서는 Backbone과 Neck 부분에 사용된 CSP 형태의 병목 연산 블록을 Transformer와 SPP [6] 모듈로 개선한 one-stage 방식의 객체 검출 네트워크를 제안한다. 제안하는 네트워크는 비전 트랜스포머(Vision Transformer, ViT) [7] 의 Multi-Head Attention을 통해 입력된 영상 전반의 feature를 학습하며 CSP 형태의 SPP 모듈과 함께 네트워크의 수용영역을 확장하여 객체 검출 성능을 개선한다.

본 논문의 구성은 다음과 같다. 2장에서는 컴퓨터 비전 영역에서 주로 사용되는 Vision Transformer와 Scaled-YOLOv4에서 제안된 CSP 블록에 대해 소개하고 3장에서는 제안하는 네트워크의 객체 검출 향상 기법과 전체적인 구조를 설명한다. 4장에서는 제안하는 네트워크의 객체 검출 성능을 평가하고 마지막 5장에서 결론을 맺는다.

2. 관련 연구

2.1 Vision Transformer

그림 1은 컴퓨터 비전 분야에서 주로 사용되는 Vision Transformer 구조이며, feature 추출 성능 향상을 위해 다수의 인코더를 중첩하여 사용한다. 동작 원리는 자연어처리 분야에서 제안된 Transformer 인코더와 유사하다. 인코더로 입력되는 영상 데이터는

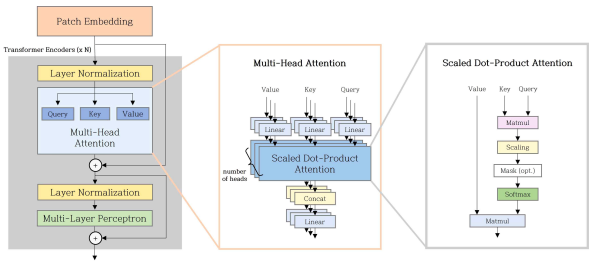


그림 1. ViT 기반의 Transformer 인코더 구조

패치 임베딩(Patch Embedding)을 통해 선형 변환된다. Patch Embedding은 입력 영상을 16 x 16의 패치 크기로 분할한 뒤, 1차원 벡터로 변환(Flatten) 한다. Transformer는 입력되는 데이터들을 순차적으로 처리하지 않고 병렬적으로 처리하기 때문에 데이터 순서에 대한 정보가 없다. 따라서 학습이 가능한 위치 임베딩 벡터를 임의로 생성하여 1차원으로 변환된 벡터에 더해주어 순서정보를 보완한다. Patch Embedding을 통해 생성된 임베딩 벡터는 층 정규화(Layer Normalization) 과정을 거친 뒤 추가적인 임베딩 과정을 통해 Query, Key, Value의 세 가지 임베딩 벡터로 변환된다. Query는 현재 패치 영상에 대한 임베딩 값, Key는 비교하고자 하는 패치 영상의 임베딩 값, 그리고 Value는 Key에 대응되는 패치 영상의 임베딩 값을 의미한다.

ViT의 Multi-Head Attention에 입력되는 세 개의 임베딩 벡터는 Self-attention 연산을 위해 동일 패치 영상으로부터 임베딩 벡터를 얻는다. 세 개의 임베딩 벡터는 여러 개의 헤드(head)로 나뉘어 병렬 처리되며, 각각의 헤드는 Scaled Dot-Product Attention 기반의 Self-attention [1] 을 수행한다. Scaled Dot-Product Attention은 식 (1)과 같다. 먼저 Query값과 Key 값의 내적을 계산한 뒤, Softmax 연산을 통해 패치 영상 간 유사도를 0에서 1 사이의 값으로 정규화 한다. Softmax 연산의 안정적인 결과 산출을 위해 두 벡터의 내적 값을 Key 값의 차원 d_k의 제곱근으로 스케일링(Scaling)한다. 계산된 값들은 Value 값과 내적을 취하여 Query와의 유사도를 나타내는 값으로 변환된다. 이후 Multi-Head Attention의 출력은 잔차 연결을 통해 초기 임베딩 값에 더해지며, Layer Normalization을 거친 뒤 Multi-Layer Perceptron 계층을 통해 객체의 feature를 추출한다. 이러한 과정을 통해 CNN 기반의 feature 학습보다 더 넓은 수용영역을 가지며 ViT는 광범위한 위치에서의 문맥적 특징(Global context feature)을 학습한다.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad (1)$$

2.2 CSP 블록

그림 2의 CSP 블록은 CSPNet 구조를 따르며 Scaled-YOLOv4에서 제안된 알고리듬이다. CSP 블록은 이전 계층으로부터 입력받은 데이터를 두 가지 경로로 분리한다. 이러한 구조는 네트워크 학습 시 중복으로 사용되는 기울기(Gradient) 정보를 제거하여 연산량과 메모리 사용량은 감소시키고 모델의 정확도와 추론 속도를 증가시킨다. YOLOv4-CSP에서는 Backbone에 그림 2 (a), Neck에 그림 2 (b)의 CSP 블록을 사용하여 네트워크의 층을 깊게 쌓을수록 발생하는 연산량 증가 문제를 해결한다.

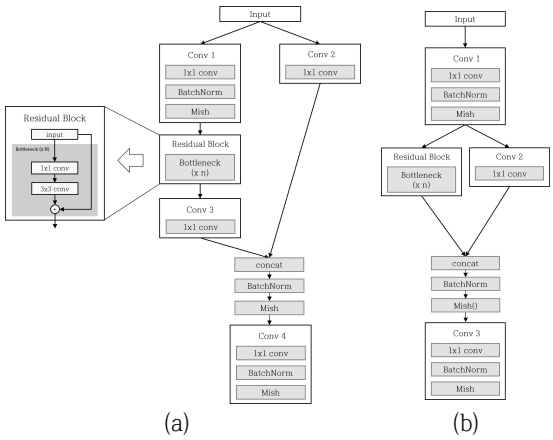


그림 2. CSP 블록

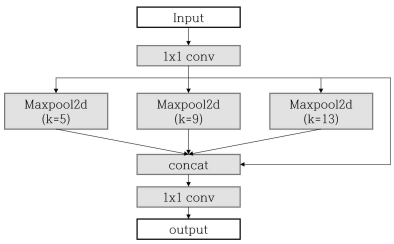


그림 3. SPP 모듈

3. 제안하는 네트워크

3.1 Transformer와 SPP 연산 기반의 CSP 블록

제안하는 네트워크는 YOLOv4-CSP의 Backbone 말단에서 추출된 feature와 Neck에서 병합되는 feature에 대해 그림 4의 CSP 블록을 사용한다. Backbone에 사용되는 CSP 블록은 기존의 병목 연산을 Transformer와 SPP 모듈로 대체한 구조이다. Transformer의 Self-attention 연산은 n개의 입력 임베딩 벡터와 각 벡터의 차원 d에 대하여 $O(n^2d)$의 계산 복잡도 [1] 를 가진다. 따라서 가장 낮은 해상도의 feature를 추출하는 계층인 Backbone 말단 부분에 그림 4의 CSP 블록을 연결한다. 제안하는 CSP 블록의 Transformer 모듈은 ViT 구조를 기반으로 하며, Backbone 말단의 입력 feature 해상도가 20x20이 되므로 Patch Embedding에서의 패치 분할 과정은 수행하지 않는다. Transformer는 4개의 헤드로 구성된 Multi-Head Attention을 통해 병렬적으로 feature를 학습하며, 학습된 feature는 CSP 형태의 SPP 모듈에 입력된다. SPP 모듈은 그림 3과 같이 입력 feature에 대해 1x1 convolution과 커널(kernel) 크기가 5, 9, 13인 최대 풀링(Max pooling) 연산을 수행한 뒤 각각의 결과를 채널별로 병합한다. Neck 부분에 사용된 CSP 블록은 기존 CSP 블록의 Residual Block에 CSP 형태의 SPP 모듈을 추가하여 수용 영역을 확장한다. 그림 4의 CSP와 CSP2는 각각 그림 2 (a)와 그림 2 (b)의 CSP 블록 구조를 따른다.

3.2 전체 네트워크 구조

전체적인 네트워크 구조는 그림 5와 같이 Backbone, Neck, Head로 구성된 one-stage 객체 검출 구조이다. 네트워크의 Backbone은 기존 YOLOv4-CSP의 CSPDarknet을 지나 입력 영상 대비 1/32배 축소된 feature를 얻고 CSP 형태의 SPP 모듈과 제안하는 알고리듬 **(A)**를 거쳐 최종 feature를 추출한다. Backbone의 CSP 블록은 신경망의 Neck은 특징 피라미드 네트워크(Feature Pyramid Network) [8] 에 상향식 경로(Bottom-up path)가 추가된 PANet [9] 구조이며 제안하는 알고리듬 **(B)**를 통해 각 층(Pyramid)의 병합된 feature를 다음 계층으로 연결한다. 네트워크의 Head는 YOLOv3 [10] 기반의 구조이며 사전에 정의된 경계 상자(Anchor box) [10] 를 바탕으로 Detect 1부터 차례대로 20x20, 40x40, 80x80 크기의 feature에서 객체를 검출한다. 검출에 사용되는 feature의 해상도가 높을수록 더 작은 객체를 검출한다.

4. 실험 환경 및 평가

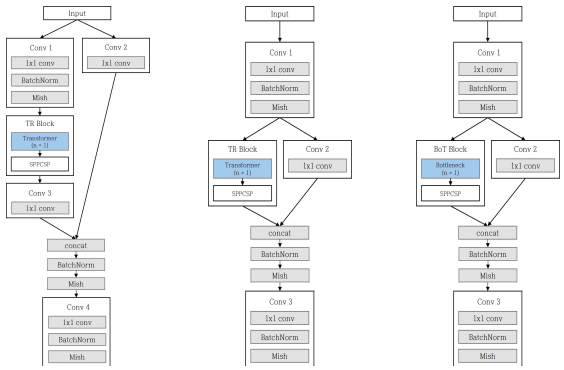
4.1 실험 환경

본 실험은 제안하는 네트워크의 학습 및 성능 평가를 위해 MS COCO train2017과 test-dev2017 데이터 셋 [11] 을 사용하였으며 학습에 사용된 GPU는 RTX 2080 Ti 1개이다. 학습 및 평가 영상 크기는 640 x 640이며 전체 학습 에포크(Epoch) 수는 100으로 설정한다. 모델의 학습 시간 및 GPU 메모리 비용 절감을 위해 YOLOv4-CSP의 경량화 모델인 YOLOv4s-mish [12] 를 사용한다. YOLOv4s-mish는 YOLOv4-CSP의 경량화 모델로 CSP 블록 수가 0.33배, 채널 수가 0.5배로 축소된 모델이다. Transformer를 제외한 모든 CSP 블록에서 mish 활성화 함수를 사용한다. YOLOv4s-mish는 모델의 하이퍼 파라미터값은 Scaled-YOLOv4 [3] 에서 사용된 값과 같다. 모델의 학습 최적화 알고리듬은 Stochastic Gradient Descent(SGD)를 사용하고 손실함수는 Generalized Intersection over Union(GIOU) Loss를 사용한다.

본 실험에서는 제안하는 객체 검출 기법의 학습 및 효율성 검증을 위해 평균 정밀도(Average Precision, AP)로 검출 정확도를 평가한다. AP는 IOU(Intersection over Union)를 기준으로 산정된다. IOU는 정답 경계 상자(Bounding box)와 학습된 모델이 예측한 경계 상자가 일치하는 비율을 의미한다. 표 1의 AP_{50}은 IOU의 임곗값을 50%로 하여 그 이상의 예측 bounding box 값을 정답으로 간주하는 평가 방식이다. 그 외에 AP_{small}, AP_{medium}, AP_{large}는 각각 정답 bounding box 크기에 따른 작은 객체, 중간 객체, 큰 객체에 대한 검출 정확도를 의미한다. 학습된 모델의 검출 정확도 및 추론 속도 측정은 RTX 1080 Ti을 사용하여 배치 크기(Batch size) 1에서 측정한다.

4.2 실험 결과

TR Block 내 Transformer 수(n)를 증가시킬수록 학습에 필요한 GPU 메모리와 연산량이 크게 증가하였으며 검출 성능 대비 추론 속도가



(a) TR-SPPCSP -CSP (b) TR-SPPCSP -CSP2 (c) BoT-SPPCSP -CSP2

그림 4. 제안하는 CSP 블록

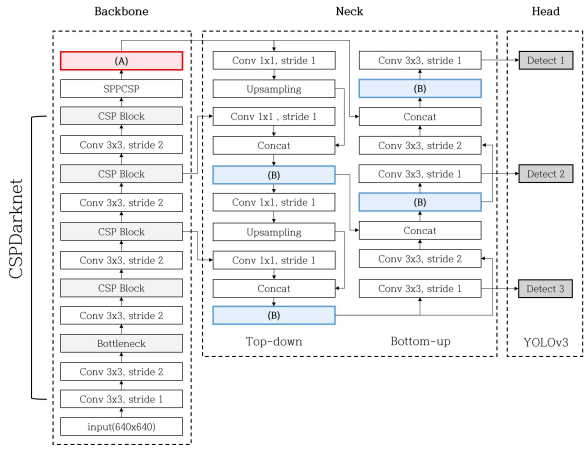


그림 5. 제안하는 네트워크 구조

급격히 감소하였다. 따라서 제안하는 네트워크의 실시간 성능을 고려하여 n값은 1로 설정하여 실험하였다.

(A) TR-CSP + (B) CSP block : (B)의 CSP block은 그림 2 (b)의 블록을 의미한다. 큰 물체에 대한 검출 정확도를 0.5% 개선하였다.

(A) TR-SPPCSP-CSP + (B) CSP block : TR-CSP 모듈에 SPPCSP 모듈을 추가하여 AP와 AP_{50}이 0.6% 향상되었으며 큰 물체에 대한 검출 정확도가 1.3%로 5가지 평가 지표 중 가장 큰 성능 향상을 보였다.

(A) TR-SPPCSP-CSP2 + (B) CSP block : (A)에 TR-SPPCSP-CSP를 사용한 모델과 비교하여 5가지 평가 지표 모두 소폭 상승하며 추론 속도가 5.6% 향상되었다. CSP2 구조가 객체 검출에 효과적인 방법임을 확인하였다.

(A) TR-SPPCSP-CSP2 + (B) SPPCSP-CSP2 : GPU 메모리를 고려하여 Batch size를 24로 조정하여 평가하였다. SPPCSP-CSP2는 CSP block의 Residual Block 내 Bottleneck을 제거한 모듈로 (B)에 CSP block을 사용한 모델 대비 AP가 1.4%, AP_{50}이 1% 향상되며 작은 객체에 대한 검출 정확도가 21%로 전체 실험 중 가장 높았다.

(A) TR-SPPCSP-CSP2 + (B) BoT-SPPCSP-CSP2 : (B)에 SPPCSP-CSP2를 사용한 모델에 비해 AP가 0.3% 향상되며 작은 객체에 대한 검출 정확도가 0.2% 감소하지만 AP_{medium}가 0.6%,

표 1. 제안하는 네트워크의 검출 성능 결과 표 (Method A와 B는 각각 그림 5의 **(A)**, **(B)**에 제안하는 CSP 블록을 적용한 실험을 의미한다.)

Method A (Backbone)	Method B (Neck)	Batch size	AP	AP_{50}	AP_{small}	AP_{medium}	AP_{large}	*GFLOPs*	*FPS*
YOLOv4s-mish		32	38%	56.6%	20%	41.9%	47.7%	21.3	48.3
TR-CSP	CSP block	32	38.1%	56.6%	19.9%	42.0%	48.2%	21.5	45.45
TR-SPPCSP-CSP	CSP block	32	38.7%	57.2%	20.1%	42.3%	49.5%	21.8	40.81
TR-SPPCSP-CSP2	CSP block	32	39%	57.5%	20.3%	42.4%	50.1%	23.2	43.1
TR-SPPCSP-CSP2	SPPCSP-CSP2	24	40.4%	58.5%	21.0%	43.7%	52.4%	26.8	34.48
TR-SPPCSP-CSP2	BoT-SPPCSP-CSP2	24	40.7%	58.5%	20.8%	44.3%	53.2%	28.9	36.36

AP_{large}가 0.8% 향상되었다. 기준 모델과 비교하였을 때 AP는 2.7%, AP_{50}은 1.9% 향상되며 전체 실험 중 가장 좋은 결과를 보였다.

5. 결론 및 향후 연구 방향

본 논문에서는 YOLOv4-CSP의 CSP 블록을 Transformer와 SPP 모듈로 개선한 one-stage 방식의 객체 검출 네트워크를 제안한다. 개선된 CSP 블록은 ViT의 Multi-Head Attention과 CSP 형태의 SPP 연산을 기반으로 입력영상에 대한 광범위한 위치의 문맥적 특징을 학습하며 네트워크의 수용영역을 확장한다. 제안하는 네트워크는 작은 객체에 대한 검출 성능 저하 없이 30fps 이상의 실시간 추론 속도를 유지하며 YOLOv4s-mish 모델 대비 AP 2.7%, AP_{50}는 1.9% 향상된 결과를 보인다. 향후 실시간 객체 검출 모델에 다수의 Transformer를 활용할 수 있는 네트워크 경량화 연구가 필요하다.

6. 참고 문헌

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, Aidan N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is All you Need,” *Advances in Neural Information Processing Systems,* vol. 30, 2017.

[2] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, “Transformers in Vision: A Survey,” *arXiv preprint arXiv:2101.01169*, 2021.

[3] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Scaled-YOLOv4: Scaling cross stage partial network,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 13029-13038, 2021.

[4] A. Bochkovskiy, C. Wang, and H. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[5] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, “CSPNet: A New Backbone that can Enhance Learning Capability of CNN,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 390-391, 2020.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904-1916, 2015.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[8] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 2117-2125, 2017.

[9] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path Aggregation Network for instance segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 8759-8768, 2018.

[10] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv:1804.02767,* 2018.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *Europ. Conf. on Comp. Vis.*, pages 740-755. Springer, 2014.

[12] K.-Y. Wong. YOLOv4s-mish. https://github.com/WongKinYiu/PyTorch_YOLOv4/tree/u5/models, 2020.