Transactions of KSAE, Vol. 29, No. 10, pp.951-958 (October, 2021)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2021.29.10.951

영상의 연속성 기반 Key 개념 도입을 통해 인식 성능을 향상시킨 딥러닝 네트워크 개선 연구

이 원 우[1)] · 최 윤 석[1)] · 유 진 우[*2)]

국민대학교 자동차공학전문대학원[1)] · 국민대학교 자동차IT융합학과[2)]

An Improved Deep Learning Network Based on Key Information Using Sequential Properties of Images

Wonwoo Lee[1)] · Yoonsuk Choi[1)] · Jinwoo Yoo[*2)]

[1)]Graduate School of Automotive Engineering, Kookmin University, Seoul 02707, Korea
[2)]Department of Automobile and IT Convergence, Kookmin University, Seoul 02707, Korea
(Received 26 April 2021 / Revised 9 June 2021 / Accepted 2 July 2021)

Abstract : In this paper, we propose a novel deep learning algorithm to improve the detector's performance in sequential images. By adopting a sliding window method that uses three consecutive images, the keys are generated by comparing the positions of the detected boxes for each of the images by using Generalized-IoU. Addition and Merge tasks were applied separately through comparison by using Complete-IoU after key generation. They have the effect of correcting the unexpected or incorrectly predicted bounding boxes. As a result, performance was improved in terms of average precision.

Key words : Deep leraning(딥러닝), Average precision(평균 정밀도), Recall(재현율), Bounding box(경계 상자), Loss function(손실 함수), Convolution(합성곱)

1. 서 론

이미지에서 물체를 감지하고 클래스를 분류하는 작업은 기존 영상 처리를 활용하던 방법에서 딥러닝을 활용한 방법으로 발전해왔다. Convolution의 도입을 통해 이미지의 구조적 의미를 보존하면서 이미지를 분류할 수 있게 되었고, 이미지 내에서 물체의 위치 감지와 그 클래스를 분류하는 것이 가능해졌다.

기존 딥러닝은 많은 연산 비용이 요구되어 사용에 어려움을 겪었지만, AlexNet[1)]의 제안 이후 딥러닝은 급격한 발전을 이루어냈고 네트워크 구조, 이미지 전처리, 후처리 등 다양한 주제로 연구가 진행되고 있다. 뿐만 아니라 의료, 운송, 항공 등 분야를 막론하고 딥러닝을 활용하는 연구 또한 활발히 진행되고 있다. 특히 자율주행 분야에선 You Only Look Once(YOLO),[9)] Single Shot Multibox-Detector(SSD)[10)]와 같은 Real time detector가 제안된 이후부터 카메라를 활용하기 위한 다양한 연구가 진행되고 있다. 예를 들어, DNN을 활용한 차량의 주행 경로를 예측하는 연구[15)]와 라이다와 카메라를 함께 활용하여 곡선 차선인식에 관한 연구[16)]가 진행되고 있다.

Tracking 분야에서도 역시 연속된 영상(Sequential images)을 대상으로 훈련된 객체 감지 네트워크[17)]들을 사용하여 물체에 ID를 부여하고 추적하는 연구가 진행되고 있다. 하지만 대다수의 딥러닝 네트워크는 사전 훈련된 Backbone에 전이학습을 적용하여 추가 레이어를 붙여 개별적인 상황의 사진의 물체를 가지고 훈련되었다. 이와 같은 네트워크들은 연속된 상황에서 사용은 가능하나, 해당 상황을 고려하고 만들어지지 않았기 때문에 추가적인 성능 개선이 가능하다.[19)] 본 논문에서는 개별 상황에서 훈련된 딥러닝 네트워크를 활용하여 속도의 감소 없이 성능을 개선할 수 있는 Stack of keys network를 제안한다.

*A part of this paper was presented at the KSAE 2021 Spring Conference
*Corresponding author, E-mail: jwyoo@kookmin.ac.kr

2. 관련 연구

연속된 상황에서 Key를 생성하기 위해선 첫 째로는 훈련된 Detector가, 두 번째로는 감지된 상자를 비교하기 위한 비교법이 필요하다. 비교법에 관한 많은 연구가 존재하기 때문에, 이를 활용하여 연구를 진행하였다.

2.1 딥러닝 네트워크

2.1.1 Backbone으로 활용되는 네트워크

AlexNet(Krizhevsky 등[1])은 이전까지 큰 성능 향상이 없었던 분류 작업에서 Top-5 오류율 15.4 %의 압도적인 성능을 보였다. AlexNet은 이미지 분류를 위해 깊은 합성곱 신경망(Deep convolutional neural network)을 사용해서 좋은 성능을 낼 수 있음을 보여주었다. VGGNet(Simonyan과 Zisserman[2])은 기존의 AlexNet보다 더 깊은 구조를 가지는 합성곱 신경망과 ReLU를 활용한 네트워크 구조를 제안하였다. 이 네트워크는 구조가 단순하고 학습이 쉽고 성능이 우수하여 다른 네트워크의 Backbone으로 많이 사용되었지만 파라미터 수가 너무 많아 메모리 사용이 많다는 문제를 가지고 있다. GoogleNet(Szegedy 등[4])은 Network in network(Lin 등[3])을 참고하여 Inception 모듈을 설계하였고, 작은 합성곱 레이어 여러 개를 한 개의 모듈로 구성하고 결과를 합치는 형태로 네트워크를 구성하였다. ResNet(He 등[5])은 딥러닝 네트워크가 구조가 깊고 넓을수록 성능이 좋아지는 대신 파라미터가 많아지고 학습이 어려워지는 문제를 Skip connection을 도입하여 해결하였다. ResNet은 기존 VGG16 네트워크보다 8배 이상 더 깊게 152 레이어까지 쌓은 앙상블 구성으로 ImageNet 태스크에서 Top-5 오류율 3.57 %을 달성하였다.

2.1.2 Real-Time Detector

Fast R-CNN[7]은 기존 R-CNN[6]의 ① Region Proposal을 추출하여 CNN 연산, ② Classification, ③ Bounding box regression으로 진행되던 방식을 Classification과 Bounding box regression을 한 번에 진행하는 방식으로 대체하여 속도를 향상시켰다. Faster R-CNN[8]에선 Fast R-CNN의 Selective search 방식으로 진행하면 시간이 오래 걸린다는 문제를 Region Proposal Network(RPN)를 도입하여 Convolution layer 내부에서 제안하는 방식으로 대체하여 해결한 Real-Time network이다.

Redmon 등[9]은 1-Stage network의 대표적인 알고리즘인 YOLO를 제안하였다. 입력 이미지를 Grid로 쪼개 각 Cell마다 Bounding box regression과 Classification을 진행하는 방식으로 속도가 매우 빠르다. Liu 등[10]은 마찬가지로 1-Stage network이며 속도가 매우 빠른 SSD를 제안하였다. SSD는 Convolution 연산을 진행할수록 Feature맵이 작아진다는 점을 이용하여 얕은 구역의 레이어에선 작은 물체를 감지하고, 깊은 구역의 레이어에서 큰 물체를 감지하는 방식을 사용하였다.

2.2 Bounding Box 비교 알고리즘

2.2.1 Loss Function

대부분의 딥러닝 네트워크는 Bounding box regression을 위한 손실 함수로 l1-norm과 l2-norm을 주로 사용하며, 파라미터로는 박스의 중심 좌표인 x, y와 박스의 입력 이미지 크기 대비 비율인 Width, Height를 사용한다. 그리고 수렴 결과는 예측한 박스의 x, y, Width, Height와 Ground truth의, x^{gt}, y^{gt}, $Width^{gt}$, $Height^{gt}$ 간의 Intersection Over Union(IoU)를 사용하여 평가한다.

Generalized-IoU(GIoU)(Rezatofighi 등[11])는 이러한 방식에 의문을 제기하여 손실 함수에 대한 파라미터로 IoU를 직접 도입하는 방식을 제안하였다. IoU의 경우 박스가 중첩되지 않은 경우에는 값이 모두 0이므로, 이를 보완하기 위해 예측한 Bounding box와 Bounding box^{gt} 두 박스를 모두 감싸는 영역의 넓이 C를 도입하여 IoU를 계산하는 방법을 제안하였다. Complete-IoU(CIoU)(Zheng 등[12])는 GIoU의 수렴 속도가 느린 점을 보완하기 위해 C 대신 두 박스의 중심점 간 거리와 두 박스 간의 비율 관계를 모두 파라미터로 도입하여 수렴 속도를 향상시켰다.

2.2.2 이미지 비교 알고리즘

Mean squared error는 가장 간단하게 이미지 유사도를 계산할 수 있는 방법이다. 하지만 픽셀 구조나 상대적 수치를 반영하기 어렵다는 문제가 있다. 이를 개선한 방식

Table 1 Structure of stack of keys network

Algorithm (Detections of Image1 and Image2 are performed before entering the while loop)
while(When there is an input value)
① Enter the next image
② Image detection using Detector (get Bboxes3, Labels3, Scores3)
③ Calculation of Bboxes1↔ Bboxes2 and Bboxes1↔ Bboxes3 comparison values using a comparison algorithm (GIoU)
④ Extract the maximum value of the calculated value, filter using preset GIoU LIMIT, and generate Key1 and Key2
⑤ calculate Key2 – Key1 to create a key vector that needs to be updated
⑥ Calculation of the similarity between for add box calculated based on the key vector and existing Bboxes2 (using CIoU)
⑦ Boxes less than the preset CIoU Criteria value are added.
⑧ Other boxes are merged with the boxes inside Bboxes2 with the highest similarity.
end

이 Structure Similarity Index Measure(SSIM)(Wang 등[13]) 이며 밝기, 대비, 구조를 모두 고려하여 이미지 유사도를 구할 수 있다. Feature Similarity Index Measure(FSIM) (Zhang 등[14])는 SSIM을 변형한 기술이며 두 이미지를 그대로 놓고 비교하면 불필요한 정보들까지도 비교하게 된다고 생각하여 저차원의 중요한 특성들을 가지고 비교하는 기술이다. 또한 해당 논문에서는 FSIM에 색상 정보를 포함하여 활용하는 방식인 FSIMc도 제안하였다.

3. Proposed Stack of Keys Network

위에서 언급한 바와 같이, 대부분의 딥러닝 네트워크는 연속된 영상에서 훈련하기보다 단일 이미지에서 물체를 감지하기 위해 학습된다. 본 논문에서는 단일 이미지를 처리하는 네트워크를 그대로 사용하여 연속된 이미지를 처리하기 위한 특화된 알고리즘을 개발하였다. 우리가 제안하는 구조는 기존 네트워크 구조를 건들지 않고 성능을 향상시킬 수 있는 후처리 알고리즘이다. 기존 네트워크를 추가 훈련 없이 성능을 향상시켜 연속된 영상에서 활용하기 위한 본 연구에선 10 Frames per second(FPS) 이상의 연속된 상황에서 물체가 영상 밖으로 서서히 나가거나 카메라로부터 멀어지는 등의 특수한 경우를 제외하면 갑자기 물체가 사라질 수 없다는 가정을 통해 물체 소실을 보정하는 방식으로 네트워크를 개선하였다. Stack of Keys Network(Table 1)는 연속된 세 개의 이미지를 슬라이딩 윈도우 방식으로 묶어서 처리하며 검출된 Bounding boxes 간의 관계를 구하여 보정하는 구조로 설계되었다. Table 1에 표시되어 있는 중앙의 점선을 기준으로 Key 생성 블록과 Add/Merge 판단 블록으로 나누어 작동하는 구조이며, FPS 손실을 최소화하면서 네트워크의 성능을 향상시킬 수 있다. 이에 대한 입증은 뒤의 실험에서 확인할 수 있다.

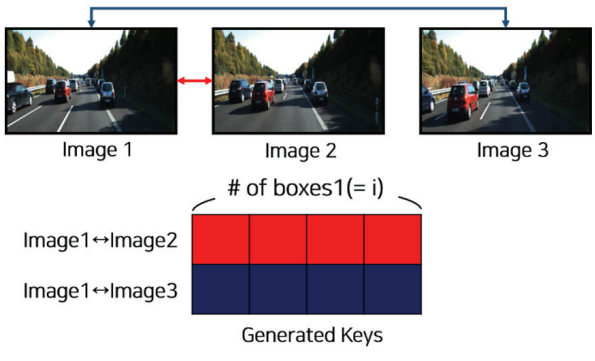

Fig. 1 The keys are created by comparing GIoU scores between bounding boxes in images. These keys are used to determine which boxes to add or merge

3.1 Key 생성 블록

딥러닝 네트워크를 활용하여 연속된 이미지를 각각 감지한다. 각 이미지 별 감지된 박스의 세트를 Bboxes라고 할 때 Bboxes1에는 i개의 박스가 생성되며, 마찬가지로 Bboxes2에는 j개, Bboxes3에는 k개의 박스가 생성된다. Bboxes1과 Bboxes2(Fig. 1에서 적색 화살표), Bboxes1과 Bboxes3(Figure 1에서 청색 화살표) 간의 관계는 아래의 GIoU[11] 식을 활용하여 수치화 하였다:

$$\mathrm{GIoU} = \mathrm{IoU} - \frac{C \setminus (A \cup B)}{|C|}$$

C는 A 박스와 B 박스를 감싸는 가장 작은 박스의 넓이이며, $C \setminus (A \cup B)$는 C 영역에서 A와 B 박스가 차지하는 영역을 제외하고 남은 영역의 넓이를 의미한다. 즉 GIoU는 -1 < GIoU ≤ 1의 범위를 갖는다. 각 박스 별 관계를 나타내는 행렬의 크기는 (i×j) 행렬, (i×k) 행렬이 되며, 각 행과 열의 최댓값들만 남긴 후 사전에 설정한 Threshold인 GIoU limit으로 필터링 한다. 이 행렬을 열을 기준으로 압축하고 연결하면 Keys를 생성할 수 있다. Key를 생성하기 위한 자세한 MATLAB 코드는 Table 2에 서술하였다. Fig. 1에서 생성된 Keys의 1행은 Image1과 Image2의 비교를 통해 생성된 Key1이며, 2행은 Image1

Table 2 MATLAB code for key generation

Key generation code

```
# Compare Bounding boxes in Image1 with Bounding boxes in Image2
   GIoU = GIoU(Bboxes1, Bboxes2)

# Extract maximum values from rows and columns
   GIoU_max_column = column_max(GIoU)
   GIoU_max_row = row_max(GIoU)

# Except for the maximum value in GIoU, all other elements are filled with 0
   for c(column number) is 1 to end
      for r(row number) is 1 to end
         if cth data of GIoU_max_column is equal to rth data of GIoU_max_row
            GIoU_key(r,c) = GIoU(r,c)
         else
            GIoU_key(r,c) = 0
         end
      end
   end

# Only elements greater than 0 and less than the threshold are converted to 1.
   GIoU = (GIoU_key > 0) and (GIoU_key < GIOU_THRESHOLD)

# In the GIoU matrix, an element with a value of 1 is replaced with a number
corresponding to each column number. the row of the GIoU matrix matches
Bboxes1 and the column matches Bboxes2.
  for i is 1 to the number of  bounding boxes1
      for j is 1 to the number of  bounding boxes2
         if GIoU(i, j) ==1
            key_sav(i, j) = j;
         else
            key_sav(i, j) = 0;
         end
      end
   end
# key_sav matrix is compressed into a one-column vector. The column number
with the index of the element 1 is stored in the created vector, the index number of
the vector is the same as the number of Bboxes1, and the data of the element is
the same as the number of Bboxes2.
   KEY1 = column_max(key_sav)
```

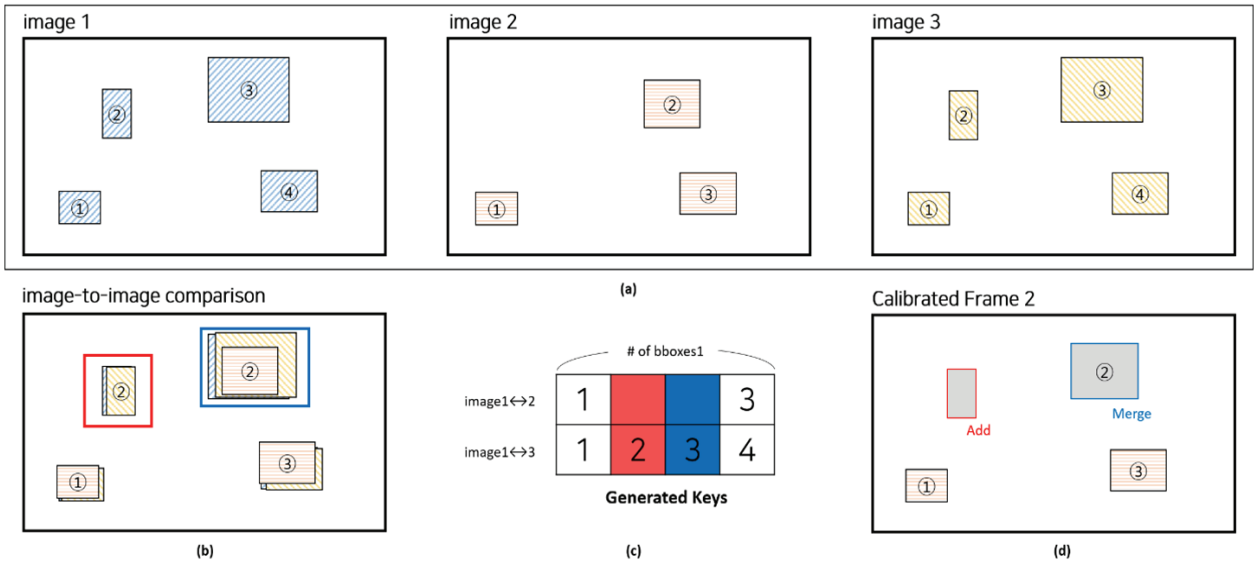

Fig. 2 (a) Comparison between images, (b) Detection of matching abnormal areas, (c) Generated keys, (d) Correction of abnormal areas using keys (Red: added box, Blue: merged box)

과 Image3의 비교를 통해 생성된 Key2이다.

생성된 Keys를 활용하면 연속된 영상에서 1번과 3번 이미지에서 감지가 된 물체가 2번 이미지에서 감지되지 않은 경우(Fig. 2(c))를 확인할 수 있다. 이러한 상황이 10 FPS 이상의 연속된 이미지에서 발생할 수 없다고 가정하였기 때문에 추가하거나 병합하는 보정작업을 수행한다.

3.2 Add/Merge 판단 블록

감지되지 않은 물체를 추가하기 전, 물체가 검출되었지만 박스의 크기나 위치정합도가 낮아서 검출되지 않았다고 판단했을 가능성을 고려해야만 한다. 이 경우에 박스를 추가하면 과검출이 발생하여 Average Precision (AP)의 하락의 원인이 된다. 이를 방지하기 위해 Key 기반으로 선정한 박스들을 추가와 병합의 Case로 분류할 수 있는 필터인 CIoU[12]를 도입하여 박스를 추가해도 되는지에 대해 판단한다.

CIoU는 IoU에서 박스 간의 거리(실제론 넓이를 사용)를 도입한 GIoU에 추가로 박스의 비율까지 파라미터로 반영하는 방식을 사용하였다. 이 때문에 GIoU보다 더 민감하게 박스 간의 관계를 정의할 수 있는 기준이 된다. CIoU[12]는 다음과 같이 계산할 수 있다:

$$\mathrm{CIoU} = 1 - \mathrm{IoU} + \frac{\rho(b, b^{gt})}{c^2} + \alpha v$$

α는 Trade-off 파라미터이며, v는 박스의 가로 세로의 비율의 연속성을 나타내기 위한 파라미터이다. 위의 식을 활용하여 추가할 박스를 Image2에 존재하는 모든 박스인 Bboxes2와 CIoU를 계산하고, 사전에 설정한 threshold인 CIoU Criteria 이상의 값이 존재하는지 검사한다. 만약 그 값이 존재할 경우, 유사한 박스가 존재하여 박스를 추가하면 안 된다고 판단하고 추가하지 않는다. 추가되지 않은 박스들은 Bboxes2에서 매칭된 박스와 병합되어 AP를 추가로 향상시키는 기능을 수행한다.

3.3 Addition/Merge Task

3.3.1 Addition Task

3.1에서 결정한 보정을 할 대상의 박스를 Key를 활용하여 Image1과 Image3에서 추출하고, 추출된 박스의 x, y, w, h의 평균 값을 Image2에 추가할 박스로 선정하였다. 3.2에서는 CIoU를 활용하여 추출된 박스에서 추가하지 않고 병합할 박스들을 걸러냈다. 이와 같은 과정 후에 걸러지지 않고 남은 박스들은 Image2에 추가(Fig. 2(d))하여 AP를 향상시킨다. 추가할 박스의 파라미터는 연산량의 증가로 인한 FPS 하락을 방지하기 위해 선형보간법을 활용하였다.

3.3.2 Merge Task

3.2에서 걸러진 박스들은 추가가 아닌 병합하는 과정을 진행한다. 박스는 가중치 없이 1~3번 이미지들에서 해당되는 박스들의 x, y, w, h 값의 평균값으로 계산되어 기존 Image2에 존재하던 박스를 대체(Fig. 2(d) 참고)한다.

Table 3 Result of comparison of mAP/FPS of reference networks and of stack of keys network. In the case of using GIoU+CIoU, it can be seen that the FPS loss is small and the AP rise is highest

Network	Backbone	Method	Reference mAP	Reference FPS	Our Algorithm mAP	Our Network FPS	improvement
YOLOv2	ResNet50	Dist + FSIMc	57.99	41	58	24	**0.02 %**
		Dist + CIoU			58.17	39.3	**0.31 %**
		IoU + FSIMc			58	10.6	**0.02 %**
		GIoU + CIoU			58.39	38.4	**0.69 %**
	ResNet101	Dist + FSIMc	72.25	31.7	72.49	20.2	**0.33 %**
		Dist + CIoU			72.62	30.8	**0.51 %**
		IoU + FSIMc			72.39	7	**0.19 %**
		GIoU + CIoU			72.63	30.3	**0.53 %**
SSD	ResNet50	Dist + FSIMc	41.61	35.2	41.74	23.1	**0.31 %**
		Dist + CIoU			41.73	29	**0.29 %**
		IoU + FSIMc			41.7	6.8	**0.22 %**
		GIoU + CIoU			41.88	33.43	**0.65 %**
	ResNet101	Dist + FSIMc	50.32	28.6	50.61	22.4	**0.58 %**
		Dist + CIoU			50.53	28.3	**0.42 %**
		IoU + FSIMc			50.47	10.3	**0.30 %**
		GIoU + CIoU			50.81	27.9	**0.97 %**

4. Experiment

4.1 실험 환경

본 연구는 MATLAB 2020b를 활용하여 진행되었으며, CPU: AMD Ryzen 7 3700X/GPU: RTX 2080ti/Ram:128gb 환경에서 실험되었다. 각각의 네트워크는 KITTI[18]에서 제공하는 데이터셋을 활용하여 훈련하였으며, 각각의 네트워크는 훈련 옵션을 *Optimizer=ADAM, LearningRate=0.0001, LearnRateDropFactor=0.01, LearnRateDropPeriod=5, MiniBatchSize=16, MaxEpochs=120*으로 설정하여 트레이닝하였다. Stack of keys network의 성능 검증은 KITTI의 Tracking datasets의 20번 데이터셋을 활용하였으며, 추적이 아닌 감지가 목적이므로 기존 Ground truth에서 감지되면 안 되는 부분(e.g. 차량에 가려져서 보이지 않는 부분)은 삭제하였다.

4.2 실험 결과

Table 3에 제안된 네트워크를 활용하여 연속된 영상을 분류한 결과를 나타내었다. Reference 네트워크로는 MATLAB에서 제공하는 YOLOv2와 SSD를 활용하였으며, Backbone은 범용적으로 활용되는 ResNet50과 ResNet101을 사용하였다. 두 블록의 필터로는 박스 간 중점의 유클리드 거리인 Dist, 감지된 박스를 Crop하여 색상 정보를 포함하여 각각의 유사도를 계산하는 FSIMc, 박스 간의 겹친 정도인 IoU, GIoU, CIoU를 조합하여 사용하였다.

실험 결과, ResNet101+YOLOv2에 GIoU+CIoU를 적용한 경우가 가장 높은 AP를 기록하였다. 추가로 FPS 손실은 4.4 % 정도에 불과하였다. 즉, 적은 FPS의 손실로 네트워크의 구성을 변경하지 않은 후처리만으로도 AP를 향상시킬 수 있다는 것을 확인하였다. 가장 높은 향상을 보인 것은 ResNet-101+SSD(0.97 %)였으며, 다른 다양한 케이스에서도 대부분 AP가 향상되었음을 확인할 수 있다.

Fig. 3은 KITTI의 10FPS로 촬영된 Tracking datasets의 20번 데이터를 활용하였으며 우측의 이미지들은 Stack of keys network를 적용한 모습이다. 황색 박스는 기존 Reference network(ResNet101+YOLOv2)를 활용하여 감지된 박스이며, 적색 박스는 감지된 결과를 토대로 계산된 추가 박스이다. 그림을 보면 기존에 감지되던 차량이 Frame 2에서는 감지되지 않은 모습을 볼 수 있는데, 제안한 네트워크가 해당 부분을 감지하여 박스를 추가한 모습을 볼 수 있다.

Fig. 4는 각 필터 알고리즘 조합을 사용하였을 때 AP의 분포를 Plot한 그래프다. (c)를 보면 GIoU와 CIoU 모두 AP에 큰 영향을 주어 마치 산과 같은 형상을 띄는 것을 볼 수 있다. 즉, GIoU와 CIoU 조합은 하강법 등을 활용하여 최적 파라미터로 수렴시킬 수 있다는 장점 또한 가지고 있다.

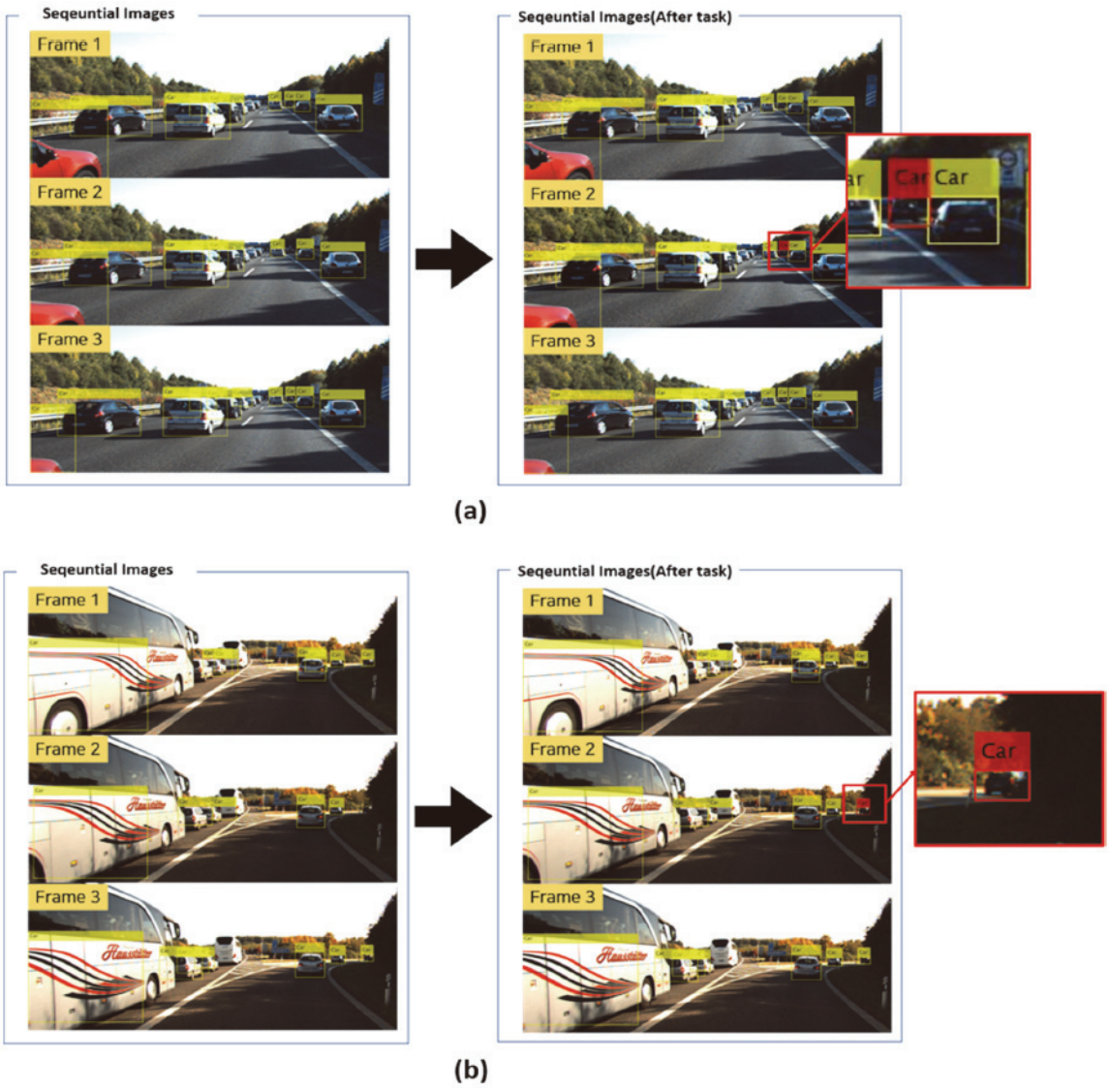

Fig. 3 A box is added to the second image using the keys created as a set of 3 of the consecutive images (left: only using the reference network/right: result of adapting stack of keys network. Red boxes were added by using keys) (a) Result of addition in case1, (b) Result of addition in case2

5. 결 론

본 논문에서는 Stack of keys network를 활용하여 추가의 훈련 없이 기존 네트워크 구조를 변경하지 않고 단순히 모듈로서 네트워크 뒤에 연결하여 후처리만으로도 성능을 향상시킬 수 있음을 보였다. 훈련에 많은 시간이 필요한 딥러닝 분야에서 추가 훈련이 필요 없다는 점은 큰 장점이다.

Stack of keys network에는 많은 연산량이 필요하지 않아 적은 FPS 손실로 AP를 향상시킬 수 있었다. 따라서 자율주행 분야 등의 빠른 동작속도를 요구하는 영역에 적합하다.

또한 차량 외의 다양한 Class에도 같은 알고리즘을 적용하여 AP를 향상시킬 수 있으며, 기존 네트워크의 성능에 상관없다는 범용성을 가지고 있기 때문에 고성능의 네트워크의 성능 역시 추가로 개선할 수 있다.

제안된 네트워크는 크게 두 블록으로 나뉘어 각각의 필터 알고리즘이 적용되었으며, 이 필터의 종류 및 특성에 따라 향상되는 성능의 수치가 달라졌다. 즉, 고성능의 새로운 필터 혹은 필터의 적용에 관한 연구가 진행되어 제안된 Stack of keys network에 결합한다면 더 높은 성능 개선을 제공하는 네트워크를 구성이 가능할 것이다.

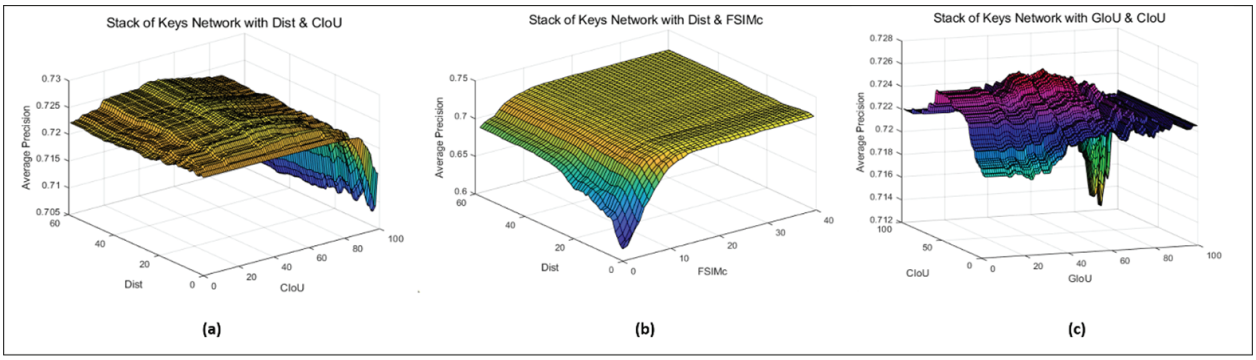

Fig. 4 A graph showing the relationship between each parameter and AP. (a) When Dist and CIoU are adopted as filter methods, (b) When Dist and FSIMc are adopted as filter methods, (c) Graph when GIoU and CIoU are adopted as filter methods

후 기

이 논문은 산업통상자원부 '산업 전문 인력 역량 강화 사업'의 재원으로 한국산업기술진흥원(KIAT)의 지원을 받아 수행된 연구(2021년 미래형자동차 R&D 전문인력 양성사업, 과제번호: N0002428)임. 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(과제번호: NRF-2021R1F1A1062153, NRF-2019R1G1A1100532).

References

1) A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, Vol.25, pp.1097-1105, 2012.
2) K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556, 2014.
3) M. Lin, Q. Chen and S. Yan, "Network in Network," arXiv preprint arXiv:1312.4400, 2013.
4) C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going Deeper with Convolutions," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.1-9, 2015.
5) K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," IEEE Conference on Computer Vision and Pattern Recognition, pp.770-778, 2016.
6) R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.580-587, 2014.
7) R. Girshick, "Fast R-CNN," Proceedings of the IEEE International Conference on Computer Vision, pp.1440-1448, 2015.
8) S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," arXiv preprint arXiv: 1506.01497, pp.1137-1149, 2016.
9) J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.779-788, 2016.
10) W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu and A. C. Berg, "SSD: Single Shot Multibox Detector," European Conference on Computer Vision, Springer, pp.21-37, 2016.
11) H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid and S. Savarese, "Generalized Intersection Over Union: A Metric and A Loss for Bounding Box Regression," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp.658-666, 2019.
12) Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye and D. Ren, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression," Proceedings of the AAAI Conference on Artificial Intelligence, Vol.34, No.7, pp.12993-13000, 2020.
13) Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE Transactions on Image Processing, Vol.13, No.4. pp.600-612, 2004.
14) L. Zhang, L. Zhang, X. Mou and D. Zhang, "FSIM: A Feature Similarity Index for Image Quality Assessment," IEEE Transactions on Image Processing, Vol.20, No.8, pp.2378-2386, 2011.
15) D. G. Jeong, M. J. Beak, W. J. Kim and S. S. Lee,

"Vehicle Trajectory Prediction based on a Deep Neural Network," Transactions of KSAE, Vol.26, No.2, pp.202-210, 2018.
16) J. C. Seo, S. T. Oh and Y. K. Kim, "A Study of Curved Lane Detection based on Dual Sensor Monitoring of LiDAR and Camera," Transactions of KSAE, Vol.29, No.2, pp.197-204, 2021.
17) D. Held, S. Thrun and S. Savarese, "Learning to Track at 100 fps with Deep Regression Networks," European Conference on Computer Vision, pp.749-765, 2016.
18) A. Geiger, P. Lenz and R. Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp.3354-3361, 2012.
19) W. Lee, Y. Choi and J. Yoo, "Improved Deep Learning Network based on Key Information for Sequential Images," KSAE Spring Conference Proceedings, p.389, 2021.