

# 2021 무인판매대 경진대회 결과 보고서

최병욱      최희웅

서울대학교 전기정보공학부

{esc5221, chw0501}@snu.ac.kr

## Abstract

무인 판매대를 실현하기 위한 필수적인 문제 중 하나는 진열대 위의 물품 배열을 각 물체의 이름과 위치를 정확하고 빠르게 *detect*하는 것이다. 이 보고서에서는 *object detection*에 뛰어난 YOLO기반 *neural network*를 통해 이 문제점을 해결하고자 하였다. 한정된 학습 이미지 개수 내에서 더 빠르고 정확도가 높은 딥러닝 모델을 학습하기 위해 *Merge*, *CutOut*, *Noise*등의 *data augmentation* 방법에 집중하였다. 가장 높은 정확도의 모델은 YOLO v4를 사용한 경우로 *processing time* 6초에 *mAP@0.5* 값 99.52를 달성하였다. 빠른 *processing time*을 중요한 평가방식으로 같이 고려했을 때는 YOLO v4-tiny를 사용한 경우로 *processing time* 5초에 *mAP@0.5* 값 98.4562를 달성하였다.

## 1. Introduction

아마존과 같은 무인 판매대를 실현하기 위해서는 사람의 이동을 추적하는 센서, 상품 진열대의 변화를 감지하는 등의 여러 단계의 각종 문제점들이 해결되어야 한다. 이 중 상품 진열대의 물체를 인식하여 진열대 위의 각 물체의 종류와 위치를 알려주는 *object detection*도 무인판매대에 꼭 필요한 기술이다. 최근 비전 분야의 딥러닝 기술이 발전하면서 10여년전에는 불가능했던 것들이 가능해졌다. 이번 보고서에서는 현재의 딥러닝 기술로 *object detection*이 어느 정도의 성능이 가능한지를 다뤘다. 널리 알려진 YOLO v3와 YOLO v4와 이를 변경시킨 *neural network*의 다양한 구조와 기본 데이터를 통한 *Merge*, *CutOut*, *Noise*등의 *data augmentation*으로 *object detection*의 성능 개선을 확인 할 수 있었다. 학습한 모델을 적용할 예정인 무인판매대에서는 *object detection*의 정확도 외에도 많은 손님이 있는 경우에 따라 모델의 빠른 반응속도도 중요하기 때문에 *detection time*(=*processing time*) 또한 중요한 평가

요소로 측정하였다. 이 보고서의 주요 핵심 사항은 다음과 같다.

- 60가지의 물체 중 진열대 위에 올라가 있는 물체들의 사진 데이터 셋 train set 103797장과 test set 345장을 제시하였으며 이를 이용하여 YOLO기반 network를 통해 *object detection* 모델을 훈련시켰다.
- *data augmentation*의 여러 종류 중 *merge*, *cutout*, *noise*등을 통해 모델의 성능 개선의 알아보았다.
- 모델의 크기에 따라 모델의 정확도와 *processing time*이 어떻게 달라지는지 확인하였으며 *processing time* 5초에 *mAP*값 98.4562를 달성한 YOLO v4-tiny 모델을 제안하였다.

## 2. Related Work

### 2.1. YOLO v3

*object detection*에 특화된 모델인 YOLO v3 [3]는 정확도와 *processing time*-측면에서 높은 성과를 보였다. 그 전의 Faster R-CNN [6]구조의 *object detection*은 ROI(Regions of Interest) 알고리즘을 사용하여 물체가 있을 법한 곳을 탐지하기 때문에 시간이 오래 걸리지만 YOLO기반의 network는 CNN을 한번에 통과시키기 때문에 속도가 빠르다는 장점이 있다.

또한 YOLO기반 *neural network*의 특징은 anchor box를 사용한다는 점이다. 이미지를 grid로 여러 구역으로 등분한 후 각 구역에서 3개의 anchor box를 생성한다. 3개의 anchor box는 서로 다른 크기를 갖기 때문에 다양한 크기의 물체를 적절하게 예측할 수 있다. 이 때 layer 깊이에 따라 3개층에서 anchor box를 예측하게 되는데 따라서 총 9가지 종류의 anchor box를 사용한다. 각 anchor box에 해당하는 output은 (중심점 x좌표, 중심점 y좌표, 너비, 높이)에 대한 수정값을 알려주는 offset과 물체가 있는지 없는지를 알려주는 정보, 물체의 종류 class개수를 포함한다.

즉,  $(grid) \times (grid)$  수 별로 (object수 + 4(offset 수) + 1)  $\times$  (number of anchor box = 3)의 output이 anchor box를 예측하는 layer 3개층에서 출력되도록 하는 모델이다. 본 문제에서는 예측해야 하는 물체의 종류가 60종류이므로 3개의 layer에서  $65 \times 3 = 195$ 개의 output이 출력이 되도록 구조를 수정하였다.

또한 backbone으로는 Darknet기반의 CNN구조를 사용했는데 전작 YOLO인 YOLO v2 [4]에서는 Darknet-19를 사용했지만 Darknet-53으로 변경하였다. 그 이유는 CNN 기반 network 중 정확도와 FPS 둘 다 충분히 빨랐기 때문이었다.

## 2.2. YOLO v4

YOLO v4 [7]는 backbone으로 CSPDarknet53 [8]으로 이용하고 Neck부분은 SPP [2], PANet [5]을 사용했으며 Head 부분은 YOLO v3를 그대로 사용하였다. 다양한 BoF(Bag-of-Freebies)와 BoS(Bag-of-Specials)를 사용하여 정확도와 속도의 향상을 보여주었다. 특히 논문에서 한 개의 1080 Ti나 2080 Ti GPU로만 학습이 가능하도록 설계한 매우 효율적인 모델이라고 소개하고 있다. 본 문제에서는 YOLO v4, v3모델과 구조는 거의 같지만 모델의 크기가 더 작은 YOLO v4-tiny 모델에 대해 정확도와 속도를 비교하였다.

## 2.3. Data Augmentation

CutOut [1] 데이터 증강 방법은 이미지의 일정 부분을 제거하는 방식이다. 제거할 때는 0으로 채우거나 랜덤한 값으로 채워넣는다. 사람은 특정 물체의 일부분이 가려지더라도 판별할 수 있는데 이러한 occlusion을 활용해 모델의 성능을 높일 수 있음을 시사하고 있다. 또한 그 외에도 training data 이미지 간의 결합을 통해 정확도를 올릴 수 있는 CutMix [9] 등의 merge data augmentation이 있다. 이를 바탕으로 object detection에서 응용할 수 있는 merge방법을 포함한 데이터 증강 방법을 적용하였다.

## 3. Method

### 3.1. Problem definition:

무인 판매대에 진열된 여러 상품들의 종류와 위치 정보 label을 detect함

실험에 사용된 물체 종류는 60종류이며 이 60종류를 활용해 촬영한 기본적인 data set은 3가지 종류이다. 첫번째 종류는 Figure 1(a)처럼 여러 상품이 진열대에 올라가 있는 상태를 측면으로 촬영한 사진 20436장이다. 두번째는 Figure 1(b)처럼 진열대 옆에서 하나의 물체를 손으로 잡고 촬영한 사진 77361장이다. 마지막으로 세번째는 Figure 1(c)



Figure 1. 60종류의 물체를 이용해 촬영 및 제작한 기본 data set



Figure 2. 진열대 정면 모습의 test set

처럼 60종류의 물체를 segmentation을 적용한 사진이다. 이러한 기본 data를 통해 Figure 2와 같은 진열대의 정면 모습안의 물체를 test set 346장에 대해 detect했을때 mAP 값을 향상시키는 것이 목표이다.

## 3.2. Merge Data(Data Augmentation)

기본 data set을 바탕으로 세 가지 방법으로 merge를 진행하여 추가 training data를 만들었다. 첫 번째와 두 번째 merge 방법은 Figure 1(c)의 data에서 물체의 bounding box를 따라 잘라낸 물체를 이용하였고 세 번째 방법은 Figure 1(b)의 data에서 똑같이 bounding box를 포함하도록 잘라낸 사진을 이용하였다. 새로 merge를 통해 생성한 이미지의 해상도는 test set의 가로 세로 비율에 맞게 640x480 또는 640x420가 되도록 설정하였다.

### 3.2.1 Random Merge

test set의 경우에는 사진의 가로 세로 비율이 640:480 또는 640:420으로 두 가지 경우였다. 640x480 또는 640x420 크기의 빈 이미지를 생성한 후, 배경을 투명하게 처리한 각 물체를 최소 2종류에서 20종류 까지 위치를 랜덤하게 설정하여 붙여넣었다. 이런 식으로 랜덤하게 이미지를 붙여넣는 경우 작은 물체가 큰 물체 밑으로 숨거나 두 물체 간의 겹친 부분이 너무 넓어서 labeling이 제대로 안 되는 현상을 관찰하였다. 따라서 각 물체의 bounding box의 중앙점이 다른 물체의 bounding box에 포함되지 않도록 제한을 주었다.

물체의 크기는 segmentation 이미지 크기의 0.4 .0.6배

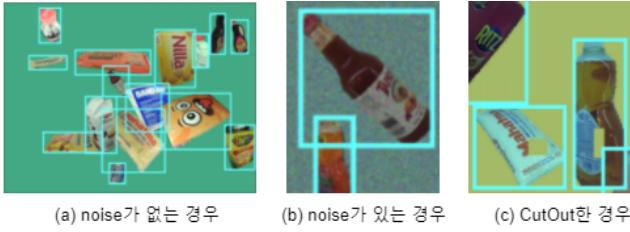


Figure 3. 15종류의 물체를 merge한 예시

사이의 다양한 크기로 랜덤하게 변경하였다. 배경은 검은색에서 흰색까지 50가지의 다양한 종류의 색깔을 랜덤하게 배정했으며 마지막에 핵심 단위로 r,g,b값을 조정하여 랜덤하게 빨강, 초록, 노랑 등의 색깔 필터를 씌워지도록 하였다. 또한 이미지에 두 가지 변화를 주었다.

- Noise: 모든 픽셀에 같은 필터를 씌우는 것이 아니라 각 픽셀 별로 랜덤한 필터를 씌웠다. noise에 강한 network를 만드는 것이 목적이다. Figure 3(a),(b)에서 noise 유무의 사진을 보여준다.
- CutOut: segmentation 이미지를 가지고 올 때 25등분한 후 그 중 한 조각을 제거하였다. 물체가 가려지는 상황에서도 인식할 수 있도록 하는 것이 목적이다. Figure 3(c)에서 CutOut된 사진을 보여준다.

정리하면 noise의 유무, CutOut의 유무에 따라 총 4가지의 다양한 경우에 따라 적게는 2종류에서 많게는 20종류의 물체를 merge하였다.

### 3.2.2 Overlapping Merge

640x480 크기의 빈 이미지를 생성한 후, 배경을 투명하게 처리한 물체를 랜덤하게 5개 선택 후, 2열로 배치하였다. 위쪽 열에는 3개, 아래쪽 열에는 2개가 오게 배치하였고, 각 열과 같은 열의 물체들은 parameter를 설정하여 일정 부분 overlapping되게 하였다. 물체의 크기 및 원본 이미지에서의 물체 bounding box의 크기가 일정하지 않아 overlapping이 잘 되지 않는 경우가 발생하여, CutOut한 이미지의 width, height 중 큰 값이 150과 175 사이의 random 값이 되도록 정규화하여 resize 후 붙여넣었다. 또한 배경색의 r, g, b값을 150과 180 사이의 random int 값으로 설정하여 배경색에 대해 일반화된 training을 시도하였다. 1차로 실행한 merge 데이터셋의 일부에서 object들의 overlapping이 과하게 발생하여 물체 bounding box 영역의 절반 이상이 가려지는 현상이 발생하였다. 이러한 데이터셋은 특징추출



Figure 4. Overlapping merge 예시



Figure 5. 비슷하게 생긴 물체 쌍

과정에 악영향을 끼쳐 학습에 악영향을 줄 수 있다고 판단하여, 과한 overlapping이 발생하지 않도록 overlapping 계수를 조정하여 merge를 실행하였다.

### 3.2.3 Grid Merge

60가지의 물체 종류의 사진을 보면 확연히 육안으로 봐도 구분이 가는 물체도 있지만 겉표지에 적혀있는 문구나 사소한 색깔 차이만 있는 물체 쌍도 존재하였다. 예를 들어 Figure 5(a)에서는 같은 크기와 문구에서 색깔만 다른 물체이고 Figure 5 (b)에서는 겉표지의 색깔만 조금 다른 소스통, Figure 5(c)에서는 표지 그림을 이루는 색 배합은 비슷하지만 그림이 다른 물체 쌍들을 볼 수 있다. 이런 물체들을 총 16가지를 추렸다. 추려진 비슷한 물체 쌍은 물체 번호로 (6, 7), (0, 3, 17, 39, 40), (16, 26), (28, 45, 46), (31, 42), (53, 54) 와 같았다.

이렇게 비슷한 물체 쌍들의 목록을 설정한 후, Figure 1(b)의 단일 물체 사진을 4장 또는 9장씩 Figure 6처럼 merge하였다. 이는 같은 train set의 단일 사진 4장 또는 9장의 효과를 한 장으로 볼 수 있기 때문에 적은 이미지로도 많은 양의 데이터를 학습이 가능하다. 또한 Figure 6(b)에서 볼 수 있듯이 앞에서 설정한 비슷한 물체 쌍들이 하나의 이미지 안에 들어가도록 하는 merge 방법도 사용해서 비슷한 물체를 구별해야 하는 상대적으로 어려운 환경에서 모델이 적응하게 하였다.

## 4. Experiments and Result

학습에 사용한 모델은 darknet YOLO v3, YOLO v3-tiny, YOLO v4, YOLO v4-tiny로 4가지였다. 학습시에는

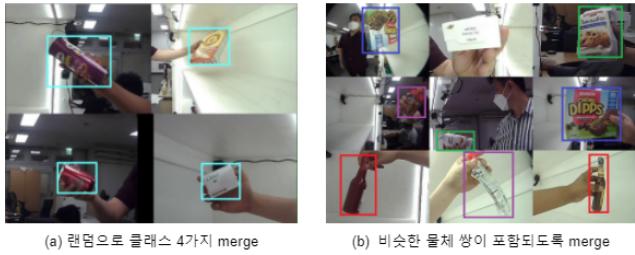


Figure 6. 단일 사진( Figure 1(b))을 4장 또는 9장씩 merge를 하였다.

Model	BFLOPS	mAP@0.5
YOLO v3	65.733	98.8
YOLO v3-tiny	5.54	92.06
YOLO v4	59.992	99.52
YOLO v4-tiny	6.8979	98.1752

Table 1. 모델 종류와 연산량 및 정확도

<https://github.com/AlexeyAB/darknet>의 코드를 사용하였다. 초기 learning rate값은 일반 모델의 경우는 0.001로 시작하였고 tiny 계열의 모델의 경우는 0.00261로 시작하였다. burn in은 1000으로 설정하여 초기에 점차 learning rate가 올라가도록 하였으며 step learning rate scheduler를 전체 iteration인 500,000의 80%, 90%에서 0.1배씩 감소하도록 설정하였다.

#### 4.1. 모델 크기에 따른 정확도 관계

기본 train set 데이터와 Random Merge와 Overlapping Merge 데이터만을 이용한 137950장의 이미지로 4가지 모델에 대해 학습을 진행하였다. 그 결과 각 모델의 최고 mAP@0.5와 각 모델에 사용되는 연산량간의 관계는 Table 1과 같았다. 가장 높은 정확도를 보여주는 모델은 YOLO v4로 mAP 99.52를 보여주었고 v3 > v4-tiny > v3-tiny 순서로 정확도가 높았다. 하지만 모델에 사용되는 연산량을 비교하면 v3보다 v4-tiny가 대략 1/10로 작지만 정확도는 대략 0.63밖에 차이가 안남을 확인하였다. 따라서 processing time을 중요한 평가요소로 생각한다면 YOLO v4-tiny가 가장 적합한 모델이라고 볼 수 있고 추후 실험은 YOLO v4-tiny를 중점적으로 이용하였다.

#### 4.2. 데이터 셋 조합에 따른 정확도 개선

주어진 train set 데이터 9만여장을 YOLO v3에서 학습한 경우 mAP 값이 64가 나왔지만 Random Merge, Overlapping Merge의 데이터를 포함하여 YOLO v3에서 학습시킨 경우 mAP 값이 91로 27이 상승했음을 확인하였다. 또한 Random Merge, Overlapping Merge에서 너무 겹쳐지

는 경우나 한 물체가 다른 물체 뒤에 숨는 경우를 제거한 데이터 셋을 이용한 경우 mAP가 98.8로 91보다 7.8이 증가함을 확인하였다. 현실에서 찍은 이미지가 아니더라도 segmentation 이미지를 조합하여 추가적인 데이터를 넣어준다면 모델의 정확도를 높일 수 있음을 알 수 있다.

YOLO v3모델에서 학습효과가 좋았던 데이터 셋에 대해 YOLO v4에서 학습한 경우는 mAP값 99.52를 달성하였고 YOLO v4-tiny에서 학습한 결과는 98.1752이었다. 추가로 Grid Merge 데이터를 포함시켜 YOLO v4-tiny를 학습시킨 결과 mAP값 98.4562를 얻었다.

#### 4.3. 잘 안된 것들

Figure 1(b)처럼 단일 사진을 모두 넣으면 오히려 정확도가 감소하는 경향을 확인하였다. 실제로 촬영한 데이터의 숫자보다 직접 제작한 데이터의 종류 및 비율이 중요하다는 것을 알 수 있다.

또한 Random Merge의 경우 많은 이미지를 합치는 경우(16 20장)의 데이터를 제거하여 학습시킨 경우 정확도가 감소함을 보았다. 너무 겹치는 부분이 많아서 학습에 방해될 것이라는 생각과 달리 오히려 학습에 도움이 되는 것을 보고 20장보다 더 많은 이미지를 merge하는 것도 고려해보는 것이 좋을 것이다.

평균적인 mAP값은 98.4562를 얻었지만 특정 class별로 mAP값을 확인한 결과 3 4가지의 물체 종류에 대해 낮은 mAP를 보이는 경향을 확인하였다. 정확도가 낮게 나온 물체가 포함된 이미지의 개수를 늘려 학습하는 것도 좋은 방법일 것이다.

## 5. Conclusion

60종류의 물체를 object detection할 수 있도록 새로운 문제에 해당하는 데이터 셋 104142장을 제시하였다. 이를 이용하여 현재까지 object detection의 속도와 정확도 측면에서 높은 성능을 보여주는 YOLO기반 deep neural network를 활용하여 학습시켰다. 그 과정에서 정확도를 높일 수 있는 data augmentation 방법 중 Random Merge, Overlapping Merge, Grid Merge을 제시하였다. YOLO v4-tiny 모델을 사용한 경우 mAP@0.5값 98.4562를 달성했으며 이는 기본 데이터 셋과 YOLO v3모델을 사용한 경우보다 대략 34만큼 높은 값이다. processing time은 5s로 다른 YOLO기반 모델에 비해 속도와 성능이 모두 상위권에 위치하였다.

## 6. What we've learned from Contest

YOLO v3, YOLO v4 및 tiny 모델 등 최신 object detection 모델을 실제 문제에 적용하는 경험을 할 수 있었다. 높은 정확도와 FPS를 보여주는 모델을 통해 실제 무인판매대에서 적용할 수 있겠다는 가능성을 확인하였다. data augmentation을 하는 과정에서 Python Library을 이용하여 이미지를 다루는 방식을 익힐 수 있었다. 또한 딥러닝 학습 과정에서 같은 모델이라면 데이터의 질 및 data augmentation을 활용한 데이터의 다양성이 모델의 성능에 결정적인 요소라는 점을 확인할 수 있었다.

## References

- [1] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1904–1916, 2015.
- [3] Redmon J and Farhadi A. A. yolov3: An incremental improvement. *In CVPR*, 2002.
- [4] Redmon J and Farhadi A. Yolo9000: Better, faster, stronger. *In CVPR*, 2016.
- [5] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 8759–8768, 2018.
- [6] 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*, 2015.
- [7] Bochkovskiy. A. Wang, C. Y, and Liao. H. Y. M. Yolov4: Optimal speed and accuracy of object detection. *CVPR*, 2020.
- [8] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, , and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. *In CVPR*, 2020.
- [9] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, , and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 6025–6032, 2019.

## Final weights path

/home/ai\_competition6/Final\_submission/yolov4-tiny-custom\_team6.weights