옛 한 글
O C R
프 로 젝 트

양진미, 편승희

CONTENTS

옛한글 OCR 프로그램

1. 개요

- 옛한글을 인식하고 분류시키는 AI 프로그램 개발
- Real-time Detection 모델 : YOLOv5, CenterNet, HRCenterNet 비교 후 성능 확인
- Classification 모델 : ResNet101, EfficientNet 비교 후 성능 확인

2. Real-time Detection

(a) YOLOv5

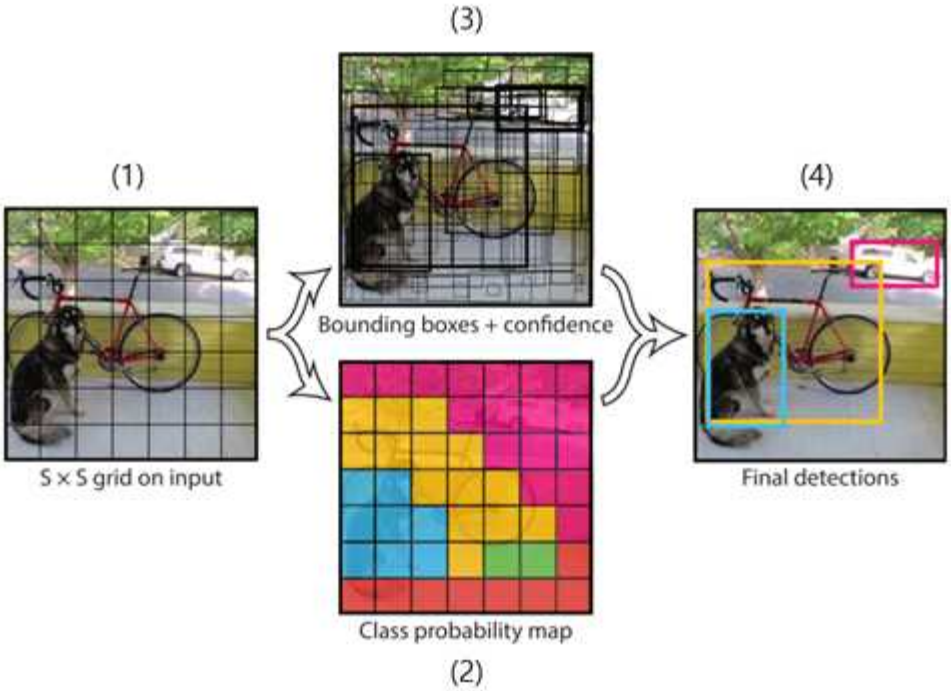


그림 2-1. YOLOv5

Yolov5는 이미지로부터 Feature Map을 생성하는 Backbone 부분에서, 4개의 conv layer를 생성하는 CSP-Darknet을 사용, Feature Map을 바탕으로 물체의 위치를 찾는 Head로 구성되어 있다.
그림 2-1과 같이 이미지를 S*S 그리드로 분할하고, 이미지 전체가 신경망을 통과하여 Bbox정보, Confidence Score 및 분류 클래스 확률 등 예측 텐서를 생성한다. 그리고 그리드 별 예측 정보를 바탕으로 Bbox를 조정 및 분류한다.

이에따라, YOLOv5의 input data는 이미지 파일과 각 이미지에 대응하는 텍스트 파일과 그에 따른 이미지 경로 텍스트 파일이 필요하고, 따라서 json 형식의 라벨링 데이터를 이용, 각각의 한자를 하나의 클래스로 통일시켜 사용했다. 또한, json 내 라벨 정보를 이용해 x center, y center를 계산했고 각 이미지 크기에 맞는 라벨 비율로 x center, y center, width, height를 수정하여 Yolov5 형식의 라벨 정보를 텍스트 파일로 저장하였다. 이미지 경로 텍스트 파일의 경우는 이미지 저장 경로를 하나의 텍스트 파일에 나열하여 생성하였다. [Yolo형식.ipynb]

(b) CenterNet

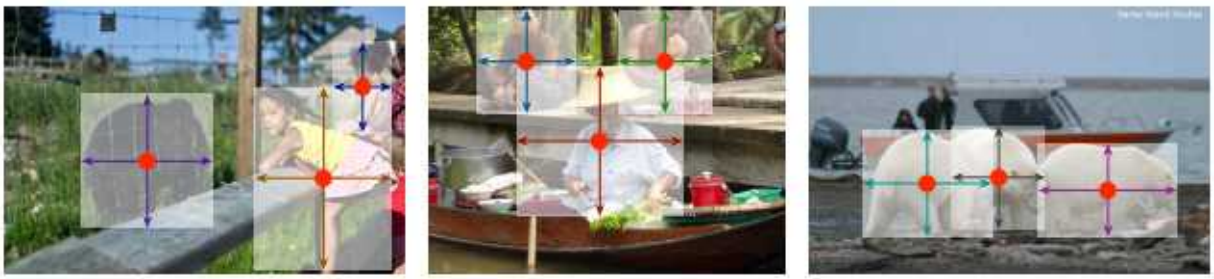


Figure 2: We model an object as the center point of its bounding box. The bounding box size and other object properties are inferred from the keypoint feature at the center. Best viewed in color.

그림 2-2. CenterNet Anchor Box

CenterNet은 그림 2-2 와 같이 Box Overlap이 아닌 오직 위치만으로 Anchor를 할당하는 것에 다른 네트워크와 차별점이 있다. CornerNet과 동일하게 단 하나의 Anchor로 Keypoint Estimation을 얻는 방법을 통해 Detection하여, 객체의 중심점 하나를 찾고 그에 따른 bounding box를 만들어낸다. 이를 통해, grouping이나 post-processing과 같은 복잡한 절차를 거치지 않게 만들었으며, 뿐만 아니라 예측된 중심점으로부터 Object size, Heatmaps, Offsets 등 다양한 정보를 regress하여 함께 bounding 한다.
CenterNet의 input data는 이미지와 각 이미지에 대응하는 coco dataset 형식의 json 파일이 하나가 필요했다. 하여, 주어진 라벨 정보를 통일 시킬 수 있도록, https://github.com/Taeyoung96/Yolo-to-COCO-format-converter 을 이용해 coco dataset 형식으로 변환시켰다.

(c) HRCenterNet

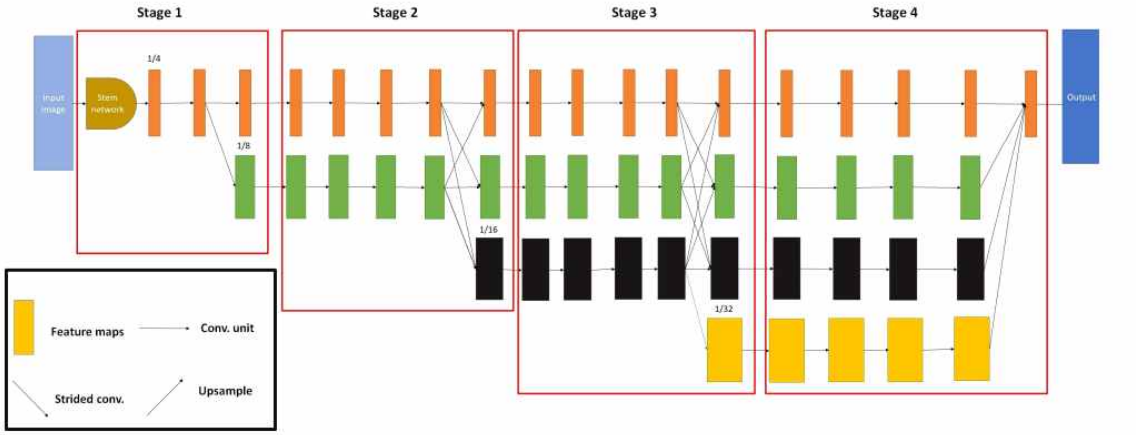


Fig 3. The architecture of our model.

그림 2-3. HRCenterNet의 구조도

HRCenterNet은 그림 2-3 처럼, CenterNet과 동일하게 하나의 Anchor로 bounding하는 Anchorless 모델을 사용한다.Object 중심에 하나의 keypoint를 찾은 후, heatmap와 높이나 넓이와 같은 관련 정보를 예측하여 바운딩 박스를 만들어 내는 방식이다. 이 모델은 한자 데이터를 위해 만들어진 모델로, 좀 더 한글 데이터 Detection에 적합할 것 같아 채택되었다.
HRCenterNet의 input data는 이미지와 각 이미지에 대응하는 라벨 정보를 담은 csv 파일이 필요하다. 이는 CenterNet과 달리 라벨 정보에 대해 top left의 x좌표, y좌표, width, height를 int형식으로 변환해야 하여, 수정한 라벨 정보를 이용하여 이미지와 라벨 정보를 담은 csv 파일을 생성하였다. [HRCenterNet형식.ipynb]

(d) Paper Benchmark

위 모델들의 논문에 같은 데이터를 바탕으로 성능이 어떻게 차이 나는지 확인 할 수 없어, 논문을 Benchmark, 그 성능을 비교했다.빠른 속도를 위해, Train data는 4386개, Validation data는 500개로 맞추었으며, 같은 데이터셋은 많이 알려진 COCO dataset을 이용하였다.

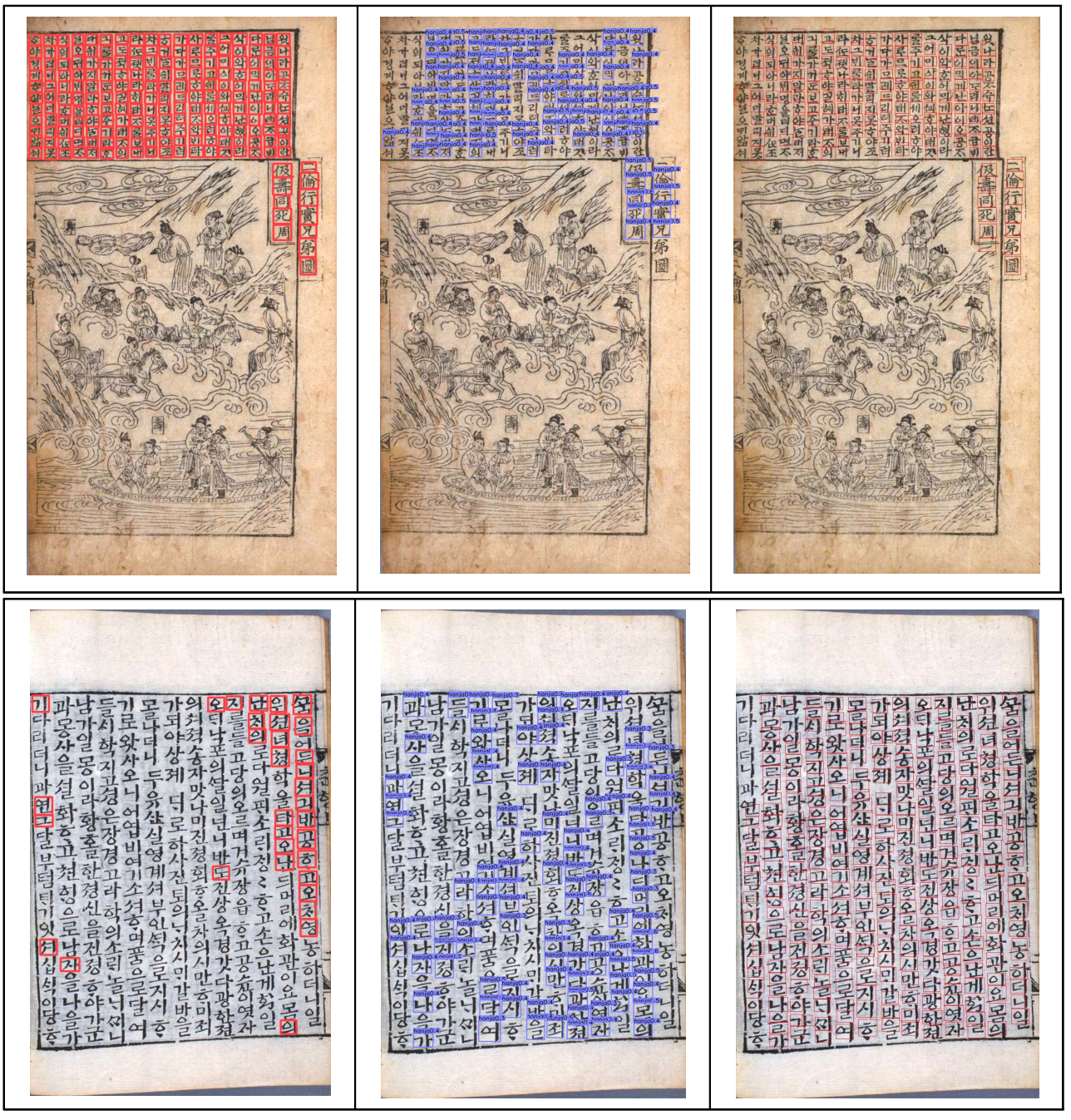
	YOLO v5	CenterNet	HRCenterNet
정확도	0.212(mAP)	0.251(mAP)	0.136(IoU)

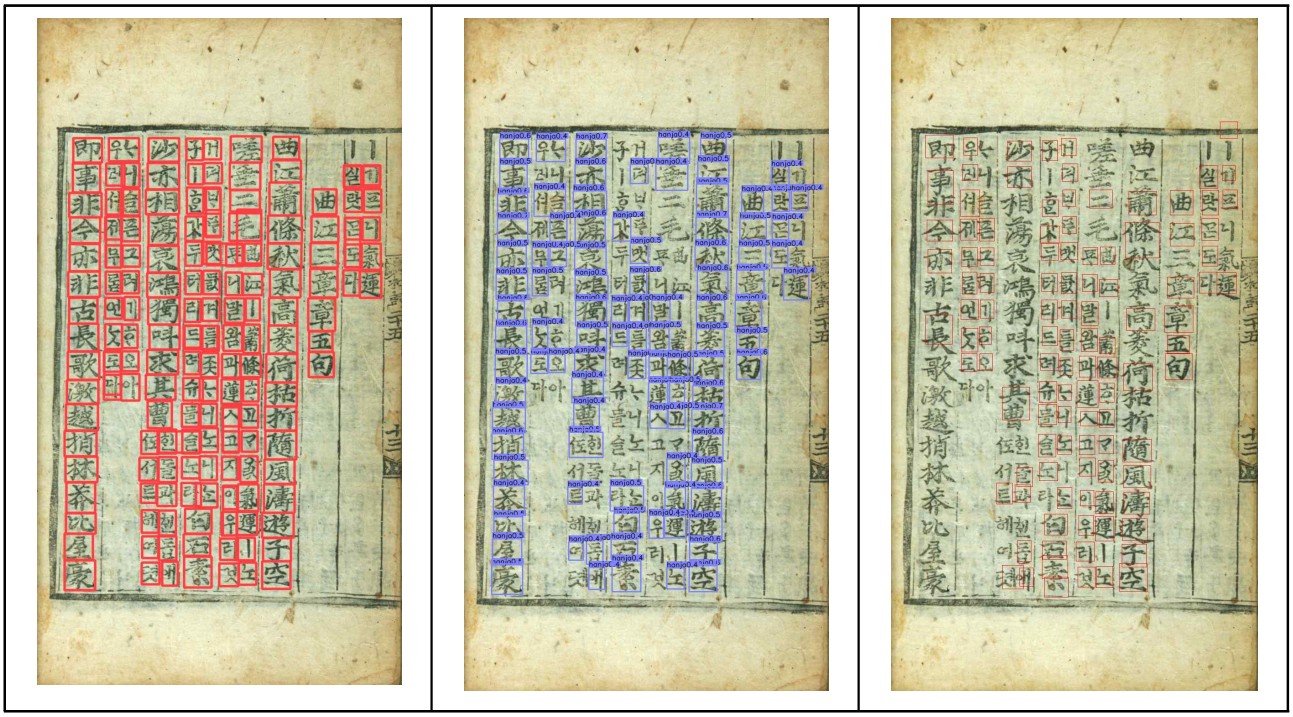
정확도는 다음과 같았으며, Test 결과는 YOLOv5, CenterNet, HRCenterNet 순이다.

(e) 한자 데이터 Train

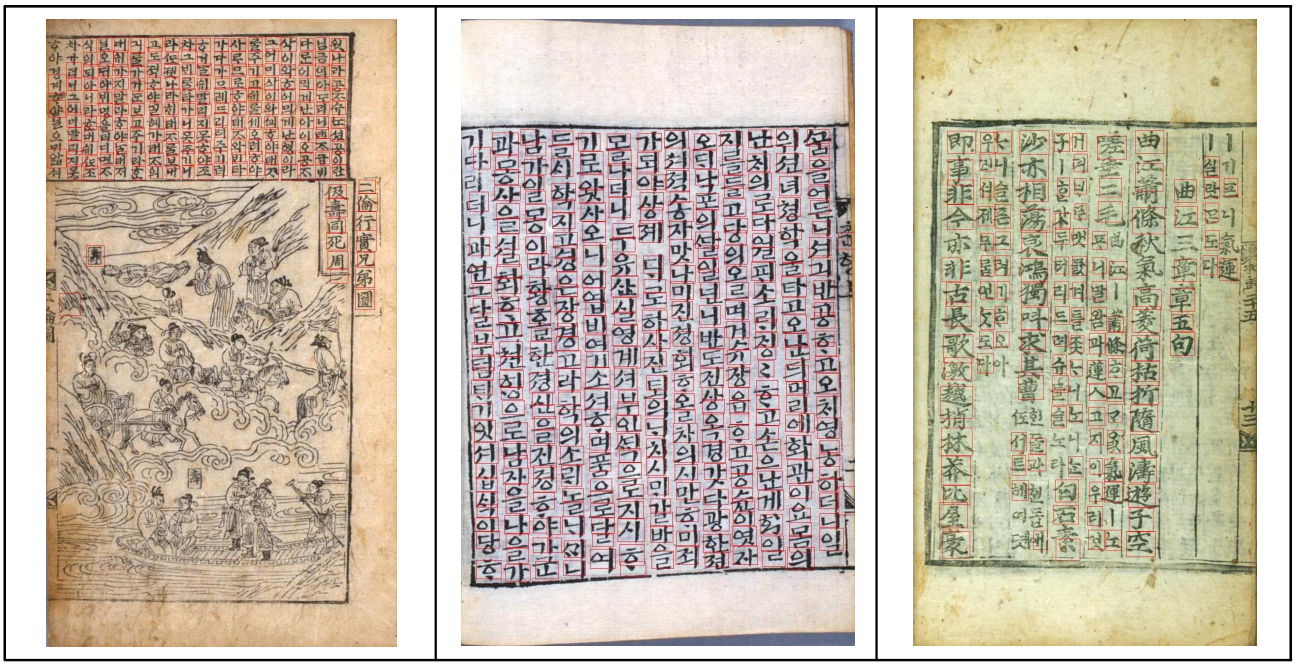
옛 한글 데이터가 아직 없어, 한자 데이터를 바탕으로 Real-time Detection 모델들의 성능을 비교하였다. 많은 데이터양으로 학습 시간이 오래 걸려, 원본 데이터의 약 40%인 Train data는 18000개, Validation data는 2273개로 맞추었으며, epoch 30, batch size 8 로 통일 시켰다.

	YOLO v5	CenterNet	HRCenterNet
정확도	81.2%	43.2%	89.2%





정확도는 다음과 같았으며, Test 결과는 YOLOv5, CenterNet, HRCenterNet 순이다. 이에 따라, HRCenterNet 모델을 epoch 8, batch size 80으로 develop 하였으며, 이에 따른 결과는 정확도 89.2(IoU)로, 다음에 첨부한 것과 같다.



3. Classification

(a) ResNet101

가장 대표적인 모델로 ResNet101 과 Efficient Net을 통해 Classification 성능을 비교하였다. 전체 데이터에서 한 글자씩 crop 후, 라벨링에 따라 나눈 뒤, Train : Validation : Test = 8:1:1 비율로 나누었으며, 이 때 한 라벨에 데이터가 0개인 라벨은 모든 데이터에서 제외하고 시작하였다.

그림 3-1. 필사체 ResNet101 Classification 결과

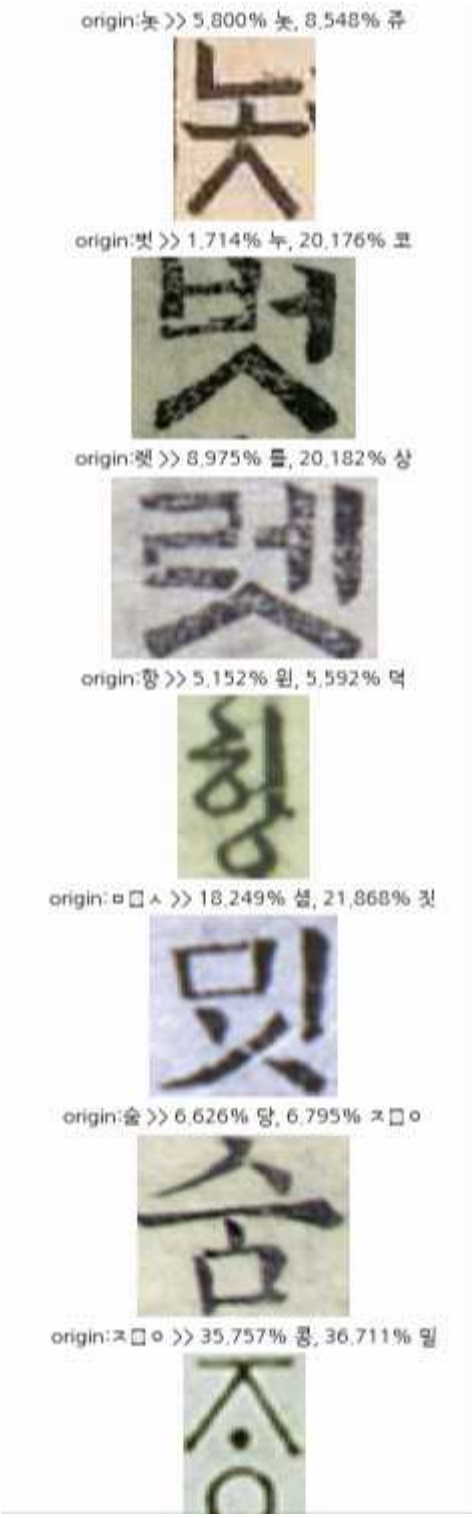


그림 3-2. 목판본 ResNet101 Classification 결과

그 결과, Classification Accuracy는 약 63%로 높았지만, 그림 3-1과 같이 Test를 진행하면 하나도 맞지 않거나

20~30개의 test 데이터에서 하나 맞는 등 높은 과적합을 보였다. 이는 Efficient Net 모델에서도 같게 높은 Accuracy에 반해 낮은 결과를 보였다. 이에 따라 데이터가 필사체는 인식하기 어려울 수도 있다 생각해, 목판본으로도 train 해보았고, 그림 3-2와 같이 똑같은 과적합 결과를 얻었다.
원인을 찾기 위해 데이터를 확인해보니, 이미지가 가장 많은 라벨링은 3~40000개 정도이나, 적은 라벨링은 1개이고, 옛한글 데이터 특성상 Flip, Rotation 등과 같은 Spatial-level Transform 이 불가하여, 너무 큰 Imbalanced data와 잘못된 라벨링(데이터의 라벨링이 잘못되어 있었음) 영향이라고 판단하였다.

(b) Solution of Imbalanced Data

잘못된 라벨링 데이터를 다시 받는 동안, Imbalanced data의 문제를 해결하기 위해 여러 해결안을 도출해보았고, 그 결과 한자 데이터를 바탕으로 crop하여 한 라벨당 이미지를 50개와 100개로 통일하여 train 해보았다. 라벨 당 50개, 100개 이상인 라벨들에서 50개, 100개씩만 random으로 뽑아내었고, 이에 따른 50개씩인 데이터는 총 라벨의 개수가 약 5000개, 100개씩인 데이터는 약 4000개였으며, 이를 Train : Validation : Test = 8:1:1 비율로 나누어 train 시켰다.

그림 3-3. ResNet101 Classification 결과

그림 3-4. Efficient Net Classification 결과

그 결과, 그림 3-3 과 3-4와 같이 ResNet101, Efficeint Net 모두 test classification 결과가 일치하였으며, Accuracy 또한 ResNet101은 94.92%, Efficient Net은 95.74%로 나왔다. 위 결과는 모두 라벨 당 데이터 50개씩인 약 5000개의 라벨로 train한 결과이며, 100개씩은 당연히 더 높은 성능을 보였고, epoch이 30임에도 높은 결과를 볼 수 있었다.

Yolov5

https://github.com/ultralytics/yolov5

1. 설치

```
$ git clone https://github.com/ultralytics/yolov5
 $ cd yolov5
 $ pip install -r requirements.txt
```

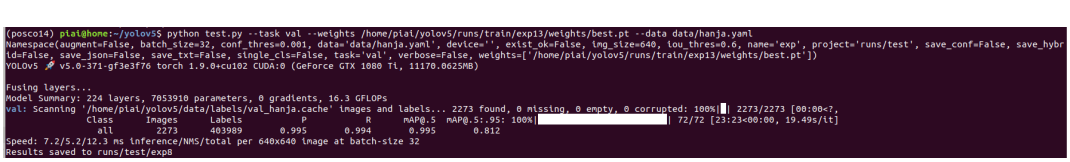
2. 학습/test/demo

a. 학습

```
$ python train.py --img 640 --batch 8 --epochs 80 --data data/hanja.yaml
 --cfg models/yolov5s.yaml --weights yolov5s.pt
```

b. test(성능확인)

```
$ python test.py --task val
 --weights /home/piai/yolov5/runs/train/exp13/weights/best.pt
 --data data/hanja.yaml
```



c. demo

```
$ python detect.py
 --weights /home/piai/yolov5/runs/train/exp13/weights/best.pt
 --source /home/piai/yolov5/data/images/test/edition
```

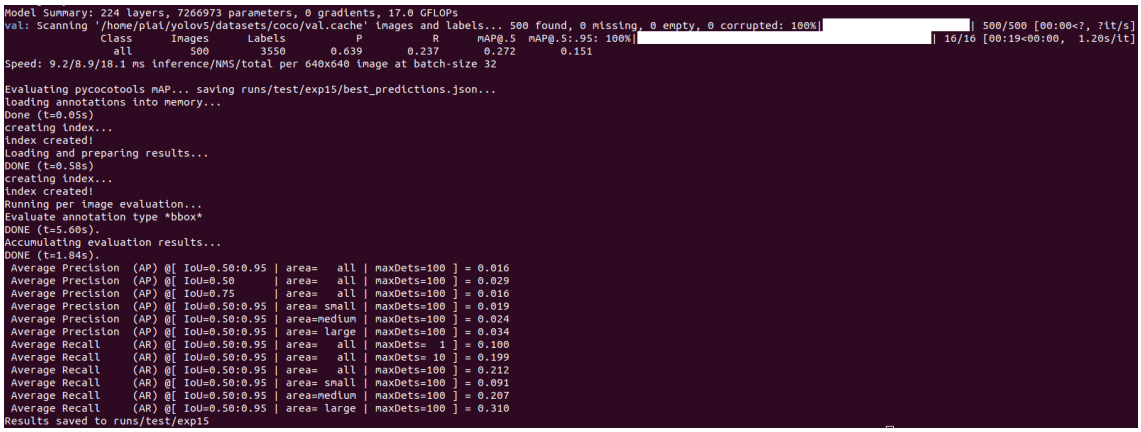
- plots.py의 plot_one_box함수에서 label 부분 주석 처리

3. COCO data-set Benchmark(train/test/detect)

```
$ java -jar cocotoyolo.jar
 /home/piai/yolov5/datasets/coco/annotations/instances_train2017.json
 /home/piai/yolov5/datasets/coco/train/images/
 "person, bicycle, car, motorcycle, airplane, bus, train, truck,
 boat, traffic light,fire hydrant, stop sign, parking meter, bench,
 bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe,
 backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard,
 sports ball, kite, baseball bat, baseball glove, skateboard, surfboard,
 tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana,
 apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake,
 chair, couch, potted plant, bed, dining table, toilet, tv, laptop, mouse,
 remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator,
 book, clock, vase, scissors, teddy bear, hair drier, toothbrush"
 /home/piai/yolov5/datasets/coco/train/labels/
```

```
python train.py --img 640 --batch 8 --epochs 60
 --data data/coco.yaml --cfg models/yolov5s.yaml --weights yolov5s.pt
```

```
python test.py --task val
 --weights /home/piai/yolov5/runs/train/exp27/weights/best.pt
 --data data/coco.yaml
```



```
python test.py --task speed
 --weights /home/piai/yolov5/runs/train/exp27/weights/best.pt
 --data data/coco.yaml
```


```
python detect.py
 --weights /home/piai/yolov5/runs/train/exp26/weights/best.pt
 --source /home/piai/yolov5/data/images/test/edition
```

2️⃣

CenterNet

https://github.com/xingyizhou/CenterNet

https://arxiv.org/pdf/1904.07850.pdf

1. my dataset → coco dataset format으로 변경

```
$ python /home/piai/CenterNet/convert/main.py
 --path /home/piai/CenterNet/data/coco/train.txt
 --output /home/piai/CenterNet/data/coco/annotations
```

5. 학습/test/demo

a. 학습

```
python main.py ctdet --exp_id coco_dla --batch_size 8 --num_epochs 30
 --master_batch 1 --lr 1.25e-4 --dataset hanja
```

b. test(성능확인)

```
python test.py --exp_id coco_dla --not_prefetch_test ctdet
 --load_model /home/piai/CenterNet/exp/ctdet/coco_dla/model_best1st.pth
 --dataset hanja
```

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.432
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.544
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.531
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.284
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.428
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.444
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.005
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.048
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.454
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.308
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.450
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.465
```

c. demo

```
python demo.py ctdet --demo /home/piai/yolov5/data/images/test/all/
 --load_model /home/piai/CenterNet/exp/ctdet/coco_dla/model_best1st.pth
```

6. COCO data-set Benchmark(train/test/detect)

- 수정 사항

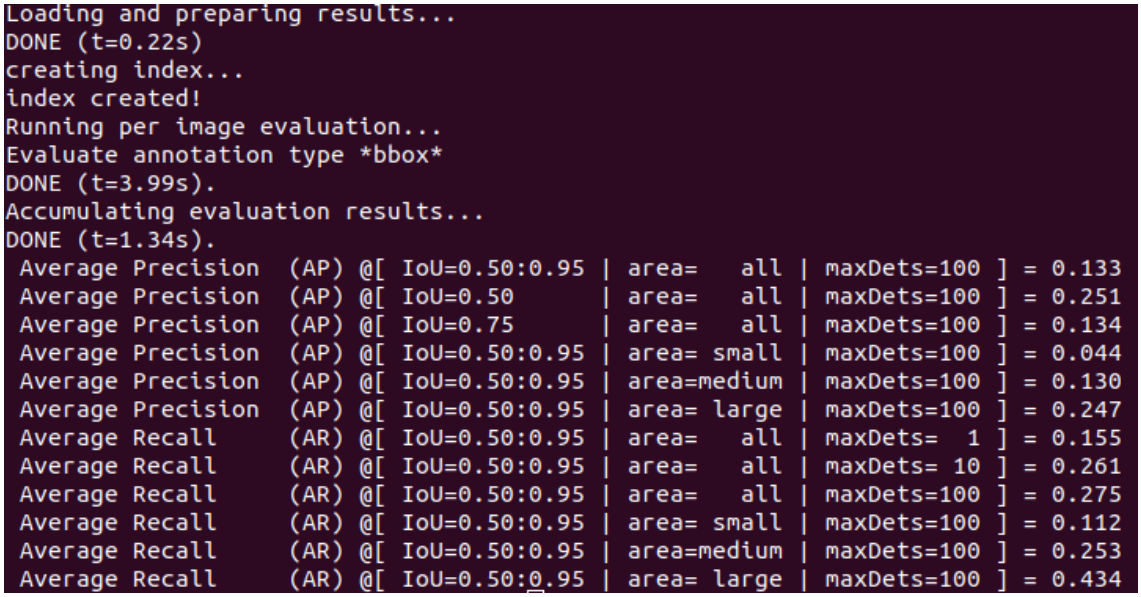
```
opts.py Line 342

 default_dataset_info = {
       'ctdet': {'default_resolution': [512, 512], 'num_classes': 80,
                 'mean': [0.408, 0.447, 0.470], 'std': [0.289, 0.274, 0.278],
                 'dataset': 'coco'},
```

- train

```
 python main.py ctdet --exp_id coco_dla --batch_size 8 --num_epochs 60
 --gpus 1 --dataset coco
```

- test

```
python test.py --exp_id coco_dla --not_prefetch_test ctdet
 --load_model /home/piai/CenterNet/exp/ctdet/coco_dla/model_best.pth
 --dataset coco
```



- demo

```
python demo.py ctdet --demo /home/piai/Documents/test/
 --load_model /home/piai/CenterNet/exp/ctdet/coco_dla/model_best.pth
```

3️⃣

HRCenterNet

https://github.com/Tverous/HRCenterNet

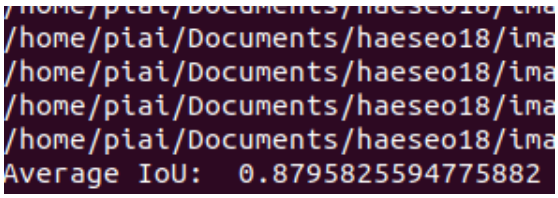
https://arxiv.org/pdf/2012.05739.pdf

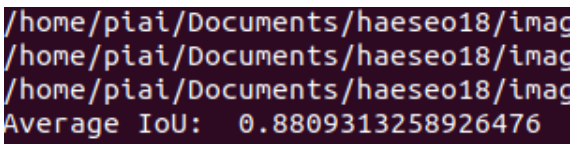
1. 학습/test/demo
    - 학습

```
python train.py --train_csv_path /home/piai/Documents/train.csv
--train_path /home/piai/Documents/trains/
--train_data_dir /home/piai/Documents/trains/
--test_path /home/piai/Documents/vals/ --test_data_dir /home/piai/Documents/vals/
--val --batch_size 8 --epoch 60
```

- test(성능확인)

```
python evaluate.py --csv_path /home/piai/Documents/haeseo18/val.csv
--data_dir /home/piai/Documents/haeseo18/images/
--log_dir weights/best.pth
```






```
python test_HRCN.py --data_dir /home/piai/AImodel/code/datasets/test_img/
```

- output

```
python test.py --data_dir /home/piai/AImodel/code/datasets/test_img/ --log_dir weights/best.pth
--output_dir /home/piai/AImodel/code/datasets/outputs/
```

```
Total TP:411025 FP:124488 FN:1213
Precision (TP/TP+FP) = 0.76754
Recall (TP/TP+FN) = 0.99706
F1-Score (2*P*R)/(P+R) = 0.86737
```

2. 수정사항

a. evaluate.py 수정

- csv_preprocess(~~args~~, args.csv_path)에서 args 제거
- 아래 코드 추가

```
if len(bbox) == 0:
        print('No object was found in the image')
        bbox.append([0, 0, 0, 0])
        score_list.append(0)
```