|  |
| --- |
| **Implementation of Tiny SSD(Single Shot Detection)** |
| 이병수(전북대학교), **임도훈(전북대학교)**, 최유진(전북대학교) |
|  |
|  |
| **요 약**  현재 컴퓨터 비전 분야에서 CNN을 이용한 많은 모델이 좋은 성과를 보이고 있으며 Object detection task도 이에 해당한다. 우리는 Object detection model 중 SSD(Single Shot Detection) model를 경량시킨 모델을 만들어 보았다.(앞으로 경량된 model을 Tiny SDD라 지칭하겠다.) Tiny SSD는 기존 논문에 쓰인 SSD model 보다 연산량과 Parameter 수를 줄인 모델로, Bounding box 개수를 줄여 연산량을 감소시키고 VGG-16 대신 VGG-11을 이용하여 Parameter size를 줄였다. Tiny SSD 모델을 학습시키기 위해 사용한 Dataset은 PASCAL VOC-2007과 VOC-2012를 사용하였고, 학습시킨 모델을 BDD100K(Berkeley DeepDrive) Dataset을 이용하여 transfer-learning(전이 학습)까지 진행하였다. |

**1. Introduction**

Object Detection Model 중 Faster R-CNN은 network를 2개로 나누어 각각 Localize와 Confidence를 학습하게 된다. 이에 반해 yolo v1과 SSD는 network를 하나만 사용하기 때문에 Faster R-CNN 보다 높은 fps를 도출한다. 하지만 yolo v1은 하나의 classifier detector를 사용하여 98개의 Bounding box를 검출하는데 속도는 빠르나 성능 면에서 Faster R-CNN 보다 좋지 않다. 반면 SSD는 6개의 classifier detector를 사용하여 8732개의 Bounding box를 검출하며 yolo v1과 달리 network가 FC layer 없이 Convolutional layer들로만 이루어져 있고 fps와 Localize, Confidence 모든 면에서 균일한 모습을 띄고 있다. yolo v1은 속도가 빠르고 Faster R-CNN은 정확도가 높은데 SSD는 전체적으로 우수한 속도와 정확도를 도출함을 볼 수 있다.

우리는 이런 SSD 모델과 축소시킨 버전인 Tiny SSD를 VOC Dataset에서 학습시킨 다음에 mAP나 fps 같은 성능 비교 지표를 사용해 성능을 비교하고, 그리고 최종적으로는 BDD100K Dataset에서 transfer-learning을 수행하는 것을 목표로 삼았다.



(그림 1.)

**2. Datasets**

PASCAL VOC-2007과 VOC-2012의 trainval imagesets을 사용하여 학습을 진행하였으며 Datasets(VOC-2007 + VOC-2012, trainval)의 크기는 16551이고 평가할 때 Datasets(VOC-2012, total imagesets - trainval)의 크기는 5585이다. 학습 기간의 기준은 epoch이 아니라 기존 논문과 같이 iteration을 이용하였고 총 120k iteration 동안 learning rate를 달리하며 학습하였다. 처음 80k iterations 동안(0 ~ 80k) learning rate는 0.0001(1e-4)를 유지하고 그 다음으로 20k iterations 동안(80k ~ 100k) 0.00001(1e-5)을, 마지막 20k iterations 동안(100k ~ 120k) 0.000001(1e-6)을 유지하며 학습을 진행했다.

학습을 완료한 모델은 BDD100K Dataset을 이용해서 transfer-learning을 진행하였다. BDD100K의 Train Datasets(size : 70000)을 사용해서 학습하였고 Val Datasets(size : 10000)을 사용해서 평가를 진행하였다. VOC와 동일하게 iteration 마다 learning rate을 달리하며 진행하고 총 50k iterations 동안 실시하였다. 처음 20k iterations 동안(0 ~ 20k) learning rate는 0.001(1e-3)을 유지하고 그 다음 20k 동안(20k ~ 40k) learning rate는 0.0001(1e-4)을, 마지막 10k iterations 동안(40k ~ 50k)은 0.00001(1e-5)을 유지하며 학습을 진행했다.

논문은 초기의 learning rate를 0.001(1e-3)로 설정하였지만 0.0001(1e-4)을 초기에 설정한 이유는 실제로 학습을 진행하였을 때, loss가 줄어들지 않는 모습을 관찰하였기 때문이다. 기본적으로 나머지 hyper parameter와 setting 값 들은 최대한 논문과 동일하게 설정하였다.(optimizer : SGD, momentum : 0.9, weight decay : 0.0005, batch size : 32)

VOC와 BDD100K의 클래스는 각각 총 20개와 10개이고 VOC의 클래스는 다양한 속성의 이미지를 가지는데 비해 BDD100K의 클래스는 Road object에 초점이 맞춰져 있다. 여기서 VOC와 BDD100K 중 겹치는 클래스는 5개로 아예 새로운 Datasets이 아니지만 어느 정도 BDD100K 클래스에 대한 정보를 알고 있기 때문에 전이 학습의 효과를 볼 수 있을 것이라 기대를 하고 진행하였고 SSD의 base-network 중 feature map을 반환하는 layer를 제외한 나머지 모두를 고정시키고 학습하였다.(loc과 conf layer는 고정시키지 않는다.)

**3. Tiny SSD**

앞서 밝혔듯이, 우리가 Tiny SSD를 구현하는데 필요한 요소로 설정한 것은 2가지로 첫 번째는 예측하는 Bounding box의 개수를 줄이는 것이고, 두 번째로는 parameter 수를 감소시키는 것이다.

기존 논문의 모델이 예측한 Bounding box 개수는 8732로 우리는 이 개수를 6792로 줄이고 싶다는 것이다. 8732개의 Bounding box가 계산되는 과정을 살펴보면 base-network(VGG-16)의 conv4\_2과 conv7 그리고 extra-network의 conv8\_2, conv9\_2, conv10\_2, conv11\_2의 6개의 feature map을 통해서 계산한다. 이때, 이 feature map들의 channel은 4 또는 6인데 이는 default box 개수와 동일하다. 즉, 아래와 그림 2. 및 식으로 8732가 도출되는 것을 볼 수 있다.

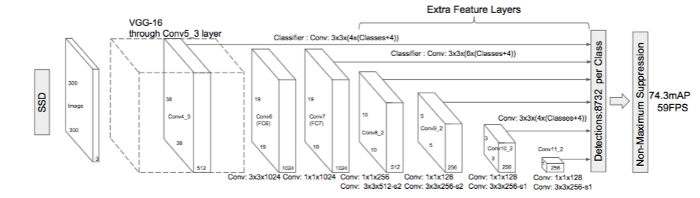


그림 2.

**(38 \* 38) \* 4 + (19 \* 19) \* 6 + (10 \* 10) \* 6 + (5 \* 5) \* 6 + (3 \* 3) \* 4 + (1 \* 1) \* 4 = 8732**

**-** ()로 감싸진 양수는 feature map의 size

위에서 4와 6은 default box 수를 의미한다고 했는데 이는 정확하지 않지만 각 feature map의 grid cell의 aspect ratio 수와 같다고 볼 수 있다. 4는 {1, 2, 1/2} 그리고 6은 {1, 2, 3, 1/2, 1/3}의 aspect ratio를 가지고 동시에 각 feature map의 grid cell scale을 계산하는데 이 scale과 aspect ratio를 이용해서 default box의 가로/세로를 구할 수 있다. 이렇게 구한 default box는 ground truth bounding box와의 Jaccard IOU가 0.5 이상일 경우 해당하는 object를 검출하는데 책임이 할당되어 학습된다. 아까 4와 6이 정확히 default box 수를 가리키지 않다고 말한 것은 실제로 4와 6에 할당된 aspect ratio는 3와 5이기 때문이고 나머지 하나는 aspect ratio가 1일 때, 작은 scale(min)을 고려하도록 설계되어있다.

여기서 우리는 aspect ratio가 1일 경우에 작은 scale(min) 또는 보통 scale(normal)을 제거해서 과연 이 부분이 얼마나 성능에 영향을 주는 지와 크기가 작은 object는 이미 feature map size가 38일 때 충분히 학습할 수 있다는 생각이 들어서 이를 알아보기 위해 Tiny SSD를 두 종류로 나누어서 학습을 시켰다. 물론 두 종류 모델 모두 예측하는 Bounding box의 개수는 6792로 같다.

두 번째로 base-network를 VGG16대신 VGG11을 사용하는 이유는 단순히 적은 parameter 수가 SSD 모델에 끼치는 영향과 모델을 조금이라도 경량화시키고자 했기 때문이다.(표 1과 표 2를 통해서 parameter의 개수 차이를 알 수 있다.) VGG16을 대신할 모델을 찾을 때 고려한 feature map size가 달라지는 부분은 모든 VGG의 conv layer(kernel = 3, stride = 1, padding = 1) 특징 때문에 고려하지 않을 수 있었다.

VGG16과 다른 점은 VGG11의 경우 base-network의 conv4\_2에서의 feature map을 사용하도록 설정한 것이다. 추가로 ResNet도 고려해보았지만 사용하지 않은 이유는 VGG와 달리 conv1의 kernel이 7로 처음부터 많은 feature들을 없애고 학습을 시작한다는 점을 들어 배제하였다.

|  |
| --- |
| base-network based on VGG16 |
| 64, 64, M, 128, 128, M, 256, 256, 256, C, 512, 512, 512, M, 512, 512, 512, M3, 1024, 1024 |
| parameters : 20483904 |

(표 1)

|  |
| --- |
| base-network based on VGG11 |
| 64, M, 128, M, 256, 256, C, 512, 512, M, 512, 512, M3, 1024, 1024 |
| parameters : 14989696 |

(표 2)

위 2개의 표는 base-network의 layer를 보여주며 숫자는 Conv kernel를 M과 C, M3는 max\_pooling의 다른 형태를 의미한다.(M = kernel : 2, stride : 2 | C = kernel : 2, stride : 2, ceil\_mode = True | M3 = kernel : 3,stride : 1, padding : 1)

**4. Experiments**

Datasets에서 밝혔듯이 모델들을 batch 크기 32와 12k iteration 동안 학습하는 조건하에 진행하였는데 Tiny SSD300(min)의 경우 batch 크기 64에서 진행하였기 때문에 6k iteration만 진행하였다.

기존 논문의 SSD300과 Tiny SSD300(normal), Tiny SSD 300(min) 모델들의 성능은 mAP와 fps 지표로 평가하였으며 사용한 테스트 Datasets은 VOC2012(total - trainval)이다. 먼저 SSD300의 mAP가 62.7로 나머지 Tiny 모델들보다 높았다. 2번째로 높은 mAP인 57.2를 기록한 모델은 Tiny SSD300(normal)이며 object인 table과 motor-bike에서는 SSD300보다 높은 성능을 보였다. 또한 Tiny SSD끼리를 비교해보면 다른 object에 비해 비교적 작은 object에 속하는 bird와 bottle에 대한 성능은 Tiny SSD(min)이 높았으며 평균적인 성능은 Tiny SSD(normal)이 좋았음을 확인할 수 있었다. 이에 대한 자세한 지표와 예측한 결과는 아래 표 3와 그림 3에서 확인할 수 있다.

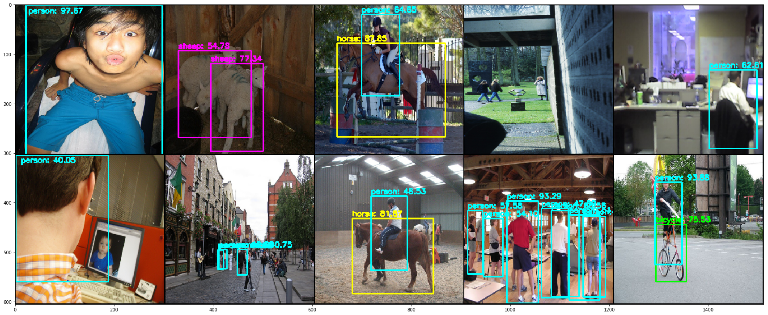
|  |  |  |  |
| --- | --- | --- | --- |
| Method | data | mAP | aero bike bird boat bottle |
| SSD300 | 12 | **62.7** | **82.3 74.5 70.1 52.1 34.3** |
| Tiny SSD300(normal) | 12 | 57.2 | 62.9 55.7 49 20.8 12.5 |
| Tiny SSD300(min) | 12 | 47.6 | 75.8 66 59.7 34.1 33.6 |

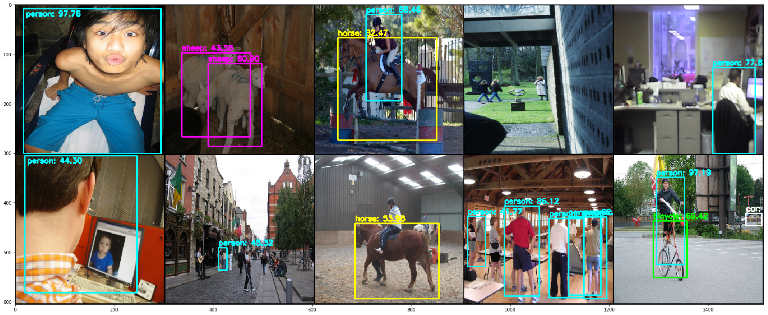
|  |
| --- |
| bus car cat chair cow table dog horse mbike person plant |
| **73.4 45.1 86 26 64.2** 64.6 **81.7 69.7** 89.6 **67.5 24.8** |
| 61 41.6 84.5 23.2 61.3 **66.3** 74.1 68.7 **89.7** 65 16.2 |
| 58 32.3 79 14.8 63.8 49 59.6 57.8 83.2 57.6 9.2 |

|  |
| --- |
| sheep sofa train tvmonitor |
| **55.3 50.9 75.1 64.4** |
| 46.6 45.5 74.9 55.5 |
| 39.9 39.1 62.3 44.9 |

(표 3)

fps의 측정결과는 SSD300보다 Tiny SSD300(normal)이 **1.34** 배 높은 것으로 확인되었고 Tiny SSD300(min)은 오히려 기존 모델보다 성능이 떨어졌다. 아마 학습이 완전하지 않은 결과로 확인되며 앞으로 확인할 필요가 있다고 생각된다.(하지만 기존 SSD 논문의 fps보다 현저히 떨어져서 이 문제도 앞으로 확인할 필요가 있다.)





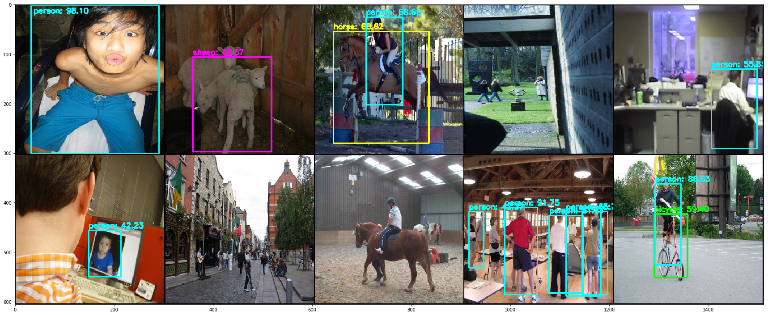


그림 3.

위부터 순서대로 SSD300과 Tiny SSD300(normal), Tiny SSD300(min) 모델에서 예측된 결과이다.

transfer-learning을 BDD100K Datasetst에서 iteration 5k 동안 진행한다고 밝혔지만 5k는 약 100시간 정도가 예상되어서 1k 동안만 학습하고 평가를 진행했다. 하지만 학습 기간이 짧은 부분과 VOC 클래스와 겹치는 클래스 인덱스 값이 일치하지 않는 점(VOC의 peraon은 14, BDD100K의 person은 4) 때문에 학습을 기대한 것만큼은 수행하지 못하는 모습을 보였다.(e.g mAP = 0.000023) 이는 학습을 진행 중도에 알게된 사실이라 어쩔 수 없지만 추후에 이를 수정해서 학습을 진행할 계획이다.





그림 4.

위부터 순서대로 SSD300에서 BDD100K Datasets의 원본과 예측한 결과이다.

**5. Conclusions**

SSD300과 Tiny SSD300(normal), Tiny SSD300min)의 성능 비교 결과는 mAP는 SSD300을 넘어설 수 없었으나 fps는 Tiny SSD300(normal)가 연산량과 parameter 수를 줄였기 때문에 우수할 수 있었다는 점을 입증할 수 있었다. 또 Tiny SSD300(normal)와 Tiny SSD300(min)을 비교해볼 때 작은 object를 감지하는 부분도 비슷비슷 했다는 점을 들어서 충분히 aspect ratio 1일 때 보통 크기의 default box로도 학습을 할 수 있다고 생각한다. 물론 두 가지의 경우에 대해 구현한 SSD300가 훨씬 우수할 것이다.

Transfer-learning 부분에서는 아쉽지만 추후에 Experiments part에서 밝혔듯이 진행해볼 계획이다. 감사합니다.