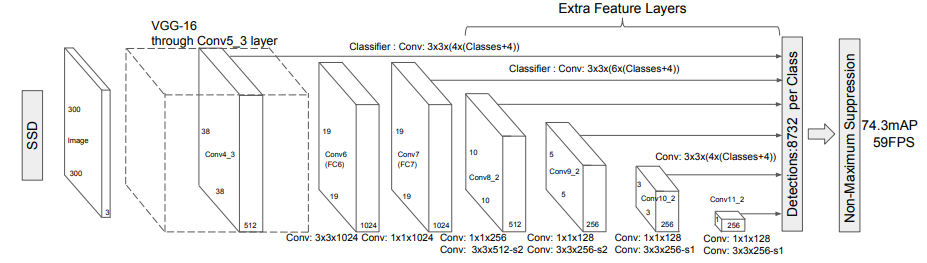
Отчет о проделанной работе

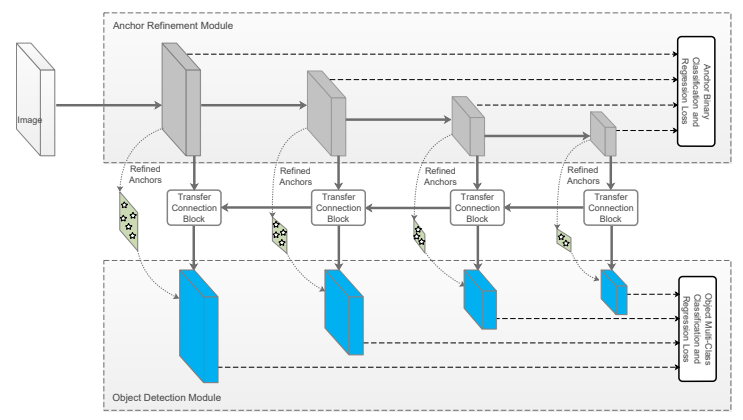
Часть 1.

Цель: изучить принципы работы сетей SSD и RefineDet, обучить сети на наборе данных Pascal VOC 07+12 так, чтобы итоговые результаты по метрике mAP на Pascal VOC 07 оказались близкими или превосходили результаты, полученные в соответствующих оригинальных статьях.

Single Shot Multibox Detector (SSD) представляет собой нейросеть, являющуюся однопроходным детектором. В основе сети лежит сеть VGG-16, в которой полносвязанные слои заменены на сверточные и в конец которой добавляются также несколько сверточных слоев [1]. Архитектуру сети можно увидеть на рис. 1.

  
Рис. 1. Архитектура сети SSD.

RefineDet также использует в основе сеть VGG-16 (хотя, как и в случае с SSD возможны и другие варианты), однако здесь добавляется только два сверточных слоя на конец сети. Процесс формирования боксов похож на тот, что используется в SSD, однако используется двухступенчатая стратегия, в которой сначала удаляются отрицательные боксы, что сокращает пространство поиска [2]. Примерную архитектуру сети можно увидеть на рис. 2.

  
Рис. 2. Архитектура сети RefineDet.

В табл. 1 приведена сравнительная характеристика вышеупомянутых сетей, а также двух других передовых детекторов.

Табл. 1. Сравнение передовых сетей, предназначенных для обнаружения объектов

|  |  |  |  |
| --- | --- | --- | --- |
| Сеть | mAP | FPS | Input Resolution |
| Faster R-CNN | 73,2 | 7 | 1000x600 |
| YOLO | 66,4 | 21 | 448x448 |
| SSD300 | 77,2 | 59 | 300x300 |
| SSD512 | 79,8 | 22 | 512x512 |
| RefineDet320 | 80 | 40,3 | 320x320 |
| RefineDet512 | 81,8 | 24,1 | 512x512 |

Как видно, SSD и RefineDet обладают как более высокими показателями FPS, так и mAP. Существует также множество модификаций SSD, таких как FFSSD, ESSD, DSSD, которые имеют либо улучшенный показатель скорости, либо точности [3, 4, 5].

Для обучения SSD было решено воспользоваться двумя версиями данной сети на фреймворке PyTorch [6, 7]. Изначально каждая из сетей была запущена на параметрах по умолчанию (learning rate = 0.001, weight decay = 0.0005, momentum = 0.9). Результат представлен в табл. 2.

Табл. 2. Сравнение полученных результатов обучения сети SSD из разных источников при параметрах по умолчанию.

|  |  |  |
| --- | --- | --- |
| class | SSD\_rep1 | SSD\_rep2 |
| aeroplane | 0,777 | 0,642 |
| bicycle | 0,8434 | 0,8058 |
| bird | 0,732 | 0,6679 |
| boat | 0,6883 | 0,5963 |
| bottle | 0,3863 | 0,3986 |
| bus | 0,8051 | 0,8088 |
| car | 0,828 | 0,6929 |
| cat | 0,8779 | 0,8721 |
| chair | 0,5637 | 0,5692 |
| cow | 0,7805 | 0,6552 |
| diningtable | 0,7256 | 0,7512 |
| dog | 0,8553 | 0,8419 |
| horse | 0,8615 | 0,8267 |
| mbike | 0,8132 | 0,785 |
| person | 0,761 | 0,6844 |
| plant | 0,4476 | 0,4362 |
| sheep | 0,7504 | 0,6404 |
| sofa | 0,7338 | 0,7926 |
| train | 0,81 | 0,8362 |
| tvmonitor | 0,7378 | 0,6745 |
| mAP | 0,7389 | 0,6989 |

Реализация с первого репозитория дала лучший результат, поэтому было решено работать далее с ней. Показатель в 74% недостаточен, поэтому для достижения более высокого результата было решено добавить класс в data augmentation, который бы отвечал за отзеркаливание изображения, а также расширены диапазоны некоторых параметров в data augmentation, ответственных за изменения фотометрических показателей входного изображения. Помимо прочего, был добавлен модуль feature-fusion по типу того, что представлен в статье Feature-Fused SSD для улучшения обнаружения мелких объектов [3]. Полученные результаты приведены в табл. 3.

Табл. 3. Результат работы сети после внесенных изменений.

|  |  |
| --- | --- |
| class | FF SSD |
| aeroplane | 0,8 |
| bicycle | 0,853 |
| bird | 0,776 |
| boat | 0,721 |
| bottle | 0,481 |
| bus | 0,852 |
| car | 0,862 |
| cat | 0,891 |
| chair | 0,594 |
| cow | 0,851 |
| diningtable | 0,763 |
| dog | 0,858 |
| horse | 0,885 |
| mbike | 0,846 |
| person | 0,781 |
| plant | 0,533 |
| sheep | 0,806 |
| sofa | 0,806 |
| train | 0,865 |
| tvmonitor | 0,778 |
| mAP | 0,7801 |

Из таблицы следует, что показатель превзошел тот, что был представлен в оригинальной статье. Зависимость суммарной ошибки по регрессии и классификации показана на рис. 3.

Рис. 3. График зависимости суммарной ошибки на оценочной выборке с ростом эпохи для измененной версии SSD.

Для проверки показателей сети RefineDet был выбран один источник [8]. После завершения обучения результат на тестовой выборке оказался на уровне 79.4%. Более подробно итоговые числа представлены в табл. 4. Данный результат считается удовлетворяющим исходной поставленной задаче, так как в дальнейшем планируется использовать данную сеть на другом наборе данных, а результат на Pascal VOC 2007 является лишь флагом того, что сеть работает должным образом.

Табл. 4. Результат работы сети RefineDet, обученной при параметрах по умолчанию на наборе данных Pascal VOC.

|  |  |
| --- | --- |
| class | RefineDet |
| aeroplane | 0,8447 |
| bicycle | 0,8566 |
| bird | 0,7709 |
| boat | 0,7141 |
| bottle | 0,652 |
| bus | 0,866 |
| car | 0,8772 |
| cat | 0,8878 |
| chair | 0,6352 |
| cow | 0,8481 |
| diningtable | 0,7536 |
| dog | 0,8581 |
| horse | 0,8695 |
| mbike | 0,8364 |
| person | 0,8212 |
| plant | 0,531 |
| sheep | 0,8028 |
| sofa | 0,8022 |
| train | 0,8835 |
| tvmonitor | 0,7734 |
| mAP | 0,7942 |

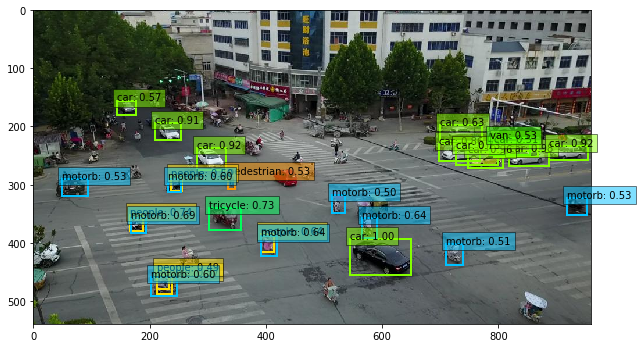
Часть 2.

Цель: изучить набор данных VisDrone2019 [9], обучить нейросеть RefineDet на данном наборе, проверить полученные результаты. Оценить работу сети при обнаружении мелких объектов.

Набор данных VisDrone представляет собой кадры из видео, записанные с дронов в различных условиях. На изображениях, как правило, присутствует достаточно большое количество объектов вплоть до трех сотен. В обучающей выборке насчитывается 6471 изображение, в тестовой выборке — 548. Всего в наборе 12 классов: игнорируемые регионы, пешеходы, люди, велосипед, автомобиль, фургон, грузовик, трицикл, крытый трицикл, автобус, мотоцикл, другое. Однако стоит упомянуть, что класс «автомобиль» присутствует гораздо чаще, нежели все другие классы.

Для работы с данным набором необходимо переписать код класса, отвечающего за преобразования разметки и файл тестирования, так как по умолчанию в исходном коде присутствует загрузчик для набора Pascal VOC. Набор Pascal характерен тем, что файлы разметки представляют собой дерево в xml-формате. Разметка в VisDrone содержится в текстовом файле формата txt, где каждая строчка файла ответственна за информацию по одному объекту класса.

Реализованные программы загрузчика и тестирования можно загрузить по ссылке (файлы visdrone.py и eval\_refinedet\_drone.py): <https://github.com/Jack55rus/VisDrone-dataloader>. Также по этой ссылке доступны примеры детектирования объектов. Один из таких примеров показан на рис. 3.

  
Рис. 4. Пример работы сети RefineDet на наборе данных VisDrone2019 при обучении на изображениях, размер которых 320 на 320 пикселей.

Сначала набор данных был обучен при размере входного изображения 320 на 320 пикселей, полученные результаты средней точности по каждому классу показаны ниже в табл. 5. Стоит заметить, что классы «игнорируемый регионы» и «другое» не участвуют в итоговой оценке. Из таблицы видно, что машины обнаруживаются точнее всего, что связано с их доминирующим количеством на всем наборе данных. Другие метрики пока не были задействованы. На официальном сайте состязания, который предлагал данный набор, можно увидеть, что лучший результат по данной метрике показала сеть DPNet, худший – SSD, набрав 54.6 и 4.78 процентов соответственно [10].

Табл. 5. Результаты работы сети на наборе данных VisDrone при разных размерах входного изображения.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Размер входного изображения | | | |
| class | 320 | 512 | 768 | 960 |
| pedestrian | 0,1296 | 0,305 | 0,3897 | 0,4088 |
| people | 0,1221 | 0,2291 | 0,2725 | 0,2831 |
| bicycle | 0,0925 | 0,1323 | 0,2073 | 0,2236 |
| car | 0,4996 | 0,717 | 0,7608 | 0,7809 |
| van | 0,2183 | 0,4337 | 0,4957 | 0,5369 |
| truck | 0,1743 | 0,3571 | 0,4291 | 0,4451 |
| tricycle | 0,1223 | 0,2461 | 0,3385 | 0,3488 |
| awningtri | 0,0625 | 0,1648 | 0,2134 | 0,1876 |
| bus | 0,2135 | 0,4725 | 0,5615 | 0,6104 |
| motorb | 0,1259 | 0,3215 | 0,4017 | 0,4275 |
| mean | 0,1761 | 0,3379 | 0,4070 | 0,4253 |

Очевидным способом улучшения результаты является увеличение размеров входного изображения, на которых обучается нейросеть.

После увеличения размеров входного изображения до 512х512, обнаружились заметные улучшения. Средняя точность по метрике AP50 выросла почти в два раза до 33.8%. Результаты представлены в третьем столбце табл. 5. Пример обнаружения на том же изображении, что показано на рис. 3, представлен на рис. 4.

  
Рис. 5. Пример работы сети RefineDet на наборе данных VisDrone2019 при обучении на изображениях, размер которых 512 на 512 пикселей.

Заметно, что результат стал намного лучше, как с точки зрения числа правильно обнаруженных объектов, так и с точки зрения уверенности по этим объектам.

При дальнейшем увеличении размеров входного изображения результат продолжает расти, однако и занимаемая память устройства возрастает тоже значительно. Так, для изображения 960 на 960 пикселей, максимальный размер батча без использования техники накоплени, на двух видеокартах с 16 Гб памяти каждая, равен 9.

Изучение работы сети на мелких объектах.

Для определения качества работы сети на мелких объектах были использованы сети, обученные на входных изображениях 512х512 и 960х960 пикселей на том же наборе данных VisDrone, в тестовых аннотациях к которому были выброшены объекты, чья площадь в пикселях превышает заданный порог. В тестовом наборе данных присутствуют изображения размеров 960х540, 1360х765, 1920х1080 пикселей. Результаты работы на метрике AP50 представлены в табл. 6-7.

Табл. 6. Результат работы сети на мелких объектах при входном изображении 512х512.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 512 | | | | | | | | | |
| class | 500 | 1000 | 2000 | 3000 | 4000 | 7000 | 10000 | 15000 | 30000 | 100000 |
| pedestrian | 0,0531 | 0,1428 | 0,2467 | 0,2777 | 0,2879 | 0,3004 | 0,3018 | 0,3037 | 0,3047 | 0,305 |
| people | 0,0722 | 0,1856 | 0,2204 | 0,2249 | 0,2268 | 0,2291 | 0,2292 | 0,2292 | 0,2291 | 0,2291 |
| bicycle | 0,0062 | 0,022 | 0,063 | 0,0907 | 0,129 | 0,1318 | 0,1323 | 0,1323 | 0,1323 | 0,1323 |
| car | 0,0492 | 0,1382 | 0,2969 | 0,4305 | 0,4871 | 0,5986 | 0,6444 | 0,6827 | 0,713 | 0,717 |
| van | 0,0062 | 0,0327 | 0,1276 | 0,1931 | 0,2302 | 0,3071 | 0,3844 | 0,417 | 0,4311 | 0,4337 |
| truck | 0,0039 | 0,0135 | 0,0352 | 0,0705 | 0,0946 | 0,1658 | 0,2188 | 0,2726 | 0,341 | 0,3559 |
| tricycle | 0,0125 | 0,0427 | 0,0985 | 0,1504 | 0,1709 | 0,2221 | 0,2417 | 0,2431 | 0,2461 | 0,2461 |
| awningtri | 0,0027 | 0,0214 | 0,0569 | 0,1344 | 0,1444 | 0,1591 | 0,1633 | 0,1648 | 0,1651 | 0,1648 |
| bus | 0,0038 | 0,0199 | 0,0493 | 0,1228 | 0,1775 | 0,2631 | 0,3044 | 0,3853 | 0,4263 | 0,4725 |
| motorb | 0,0565 | 0,1367 | 0,2736 | 0,2996 | 0,3117 | 0,3199 | 0,3209 | 0,3209 | 0,3215 | 0,3215 |
| mean | 0,0266 | 0,0756 | 0,1468 | 0,1995 | 0,2260 | 0,2697 | 0,2941 | 0,3152 | 0,3310 | 0,3378 |

Табл. 7. Результат работы сети на мелких объектах при входном изображении 960х960.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 960 | | | | | | | | | |
| class | 500 | 1000 | 2000 | 3000 | 4000 | 7000 | 10000 | 15000 | 30000 | 100000 |
| pedestrian | 0,1081 | 0,2349 | 0,3591 | 0,3893 | 0,3972 | 0,4051 | 0,407 | 0,4083 | 0,4088 | 0,4088 |
| people | 0,1018 | 0,2094 | 0,2718 | 0,2785 | 0,2805 | 0,2831 | 0,2833 | 0,2832 | 0,2831 | 0,2831 |
| bicycle | 0,019 | 0,0833 | 0,1731 | 0,2008 | 0,2158 | 0,2236 | 0,2242 | 0,2242 | 0,2236 | 0,2236 |
| car | 0,0756 | 0,193 | 0,373 | 0,4799 | 0,548 | 0,6595 | 0,7103 | 0,7524 | 0,7779 | 0,7809 |
| van | 0,0197 | 0,074 | 0,2046 | 0,2887 | 0,3336 | 0,4238 | 0,4813 | 0,5175 | 0,5341 | 0,537 |
| truck | 0,0131 | 0,0359 | 0,0821 | 0,1437 | 0,1782 | 0,2464 | 0,3089 | 0,3611 | 0,4309 | 0,4453 |
| tricycle | 0,0253 | 0,091 | 0,1932 | 0,2558 | 0,2905 | 0,3299 | 0,3469 | 0,3478 | 0,3488 | 0,3488 |
| awningtri | 0,0172 | 0,055 | 0,1119 | 0,1452 | 0,1632 | 0,1816 | 0,1856 | 0,1879 | 0,1879 | 0,1876 |
| bus | 0,0184 | 0,0782 | 0,1451 | 0,2557 | 0,3399 | 0,4228 | 0,4549 | 0,5373 | 0,5734 | 0,6104 |
| motorb | 0,0897 | 0,2189 | 0,3595 | 0,4008 | 0,4182 | 0,4265 | 0,4278 | 0,4278 | 0,4276 | 0,4275 |
| mean | 0,0488 | 0,1274 | 0,2273 | 0,2838 | 0,3165 | 0,3602 | 0,3830 | 0,4048 | 0,4196 | 0,4253 |

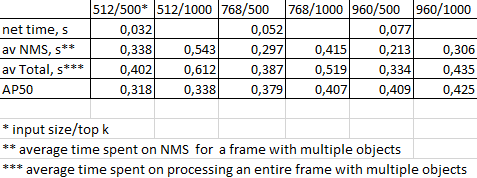
Итоговое сравнение показано на рис. 6.

Рис. 6. Зависимость метрики AP50 от максимальной площади объекта.

Как видно из табл. 6-7, многие классы (люди, велосипед, пешеход, трицикл) выходят в насыщение, начиная с определенного значения, в то время как другие (машина, автобус, фургон) продолжают расти. Это связано с тем, что первая группа классов представляет собой те объекты, которые в большинстве своем мелкие в данном наборе данных. Т.е. число объектов этого класса, превышающих порог, становится все меньше. Среднее же значение растет очень быстро вплоть до порога в 10,000, где достигает 87% от итогового значения. Далее рост идет намного медленнее. Графическое представление таблиц 6-7 представлено в приложении.

Исследование работы RefineDet при работе с видео.

Для исследования работы сети было взято похожее видео с дрона с большим количеством мелких объектов. Были протестированы сети, обученные на изображениях с размером 512х512, 768х768, 960х960 пикселей и с параметрами top\_k, равными 500 и 1000, для non-maximum suppression. Результаты показаны на рис. 7. Стоит заметить, что несмотря на увеличение качества работы сети, скорость работы сети также падает, давая FPS в 19 и 13 для 768 и 960 соответственно. Параметр net time показывает время работы сети за вычетом времени, которое тратится на non-maximum suppression. Параметр av Total указывает на то, сколько времени было затрачено всего на перевод изображения в тензор, на время работы самой сети, на работу алгоритма non-maximum suppression, а также отрисовку обрамляющих прямоугольников.

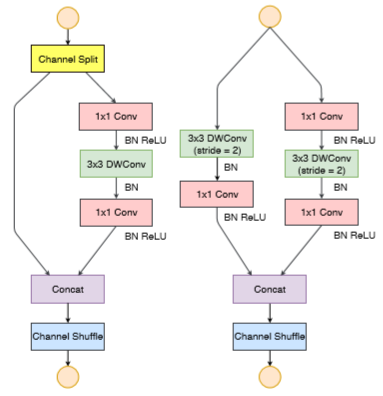
  
Рис. 7. Оценка скорости работы сети при разных размерах входного изображения и параметров non-maximum suppression.

Часть 3.

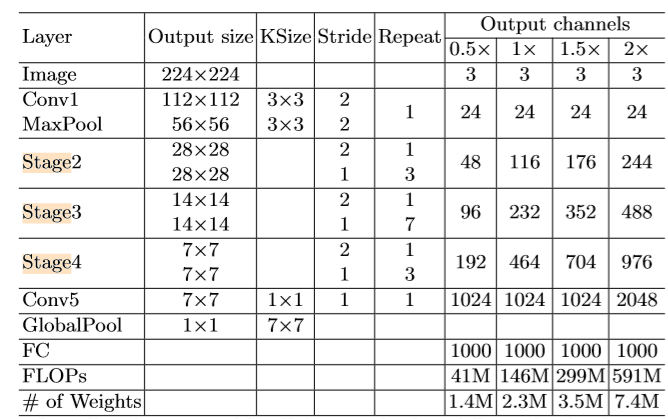
Цель: изменить базовую сеть в RefineDet с VGG-16 на более быструю, изучить возможные варианты замен базовой сети, разобраться в принципе их работы.

Как стало ясно из предыдущей части, для хорошей работы сети на мелких объектах необходимо увеличивать размер входного изображения, однако скорость работы сети при данных условиях мала. Для увеличения скорости работы сети предлагается заменить базовую сеть на более быструю. В качестве таковой была выбрана сеть ShuffleNetV2 [11]. Данный выбор основывается на соотношении качества работы и скорости [11, 12], а также доступности реализации [13].

Базовые блоки сети показаны на рис. 8.

  
Рис. 8. Базовые блоки сети ShuffleNetV2.

Общая архитектура приведена на рис. 9.

  
Рис. 9. Архитектура ShuffleNetV2

Для совместной работы с RefineDet необходимо убрать последний слой глобального пулинга и полносвязанный слой. Чтобы соблюсти размеры карт признаков, необходимо пускать на ARM-блок выходы слоев, начиная со Stage2. Для начала было решено использовать 3 слоя для обнаружения при размере входного изображения 768. Однако после 250 эпох суммарные потери примерно равны потерям для комбинации VGG16+RefineDet при входном изображении 320 после 30 эпох. Возможно, это объясняется тем, что первый идущий на обнаружение слой, который ответственен за обнаружение самых мелких объектов, в VGG находится глубже. Для решения этой проблемы предлагается брать слои, начиная со stage3, добавив свертку на конец сети, а для согласования размеров карт признаков, добавить обратные свертки. Результат работы такой конфигурации на наборе данных VisDrone приведен в табл. 8.

Табл. 8. Результат работы сети с добавлением обратных сверток и размере входного изображения 768х768.

|  |  |
| --- | --- |
| class | 768 |
| pedestrian | 0,2328 |
| people | 0,2007 |
| bicycle | 0,1142 |
| car | 0,6686 |
| van | 0,3707 |
| truck | 0,2549 |
| tricycle | 0,2298 |
| awningtri | 0,1093 |
| bus | 0,4185 |
| motorb | 0,2796 |
| mean | 0,2879 |

Как видно, результат даже немного проигрывает комбинации VGG+RefineDet для изображения 512х512. Для оценки работы сети по скорости, был выбран тот же видеоряд, что и ранее. Если сравнивать с результатом, полученным для сети с VGG и изображениях размером 768х768, приведенным на рис. 7, то данная сеть работает чуть быстрее: примерно на 12 мс. Такой результат сопоставим со скоростью работы сети при изображении размером 512х512 пикселей, но, как уже было сказано, проигрывает ему в точности. Дальнейшими шагами будут изменения конфигурации сети таким образом, чтобы точность была сопоставима, а время работы меньше, чем для соответствующих размеров изображения для комбинации VGG+RefineDet.

Приложение

Рис. П1. Графическое представление качества обнаружения мелких объектов сети RefineDet512.

Рис. П2. Графическое представление качества обнаружения мелких объектов сети RefineDet960.

Список источников

1. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg: SSD: Single Shot MultiBox Detector (2016)
2. Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, Stan Z. Li: Single-Shot Reﬁnement Neural Network for Object Detection (2017)
3. Guimei Cao, Xuemei Xie, Wenzhe Yang, Quan Liao,Guangming Shi, Jinjian Wu: Feature-Fused SSD: Fast Detection for Small Objects (2017)
4. Liwen Zheng, Canmiao Fu, Yong Zhao: Extend the shallow part of Single Shot MultiBox Detector via Convolutional Neural Network (2017)
5. Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, Alexander C. Berg: DSSD: Deconvolutional Single Shot Detector
6. <https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection>
7. <https://github.com/amdegroot/ssd.pytorch>
8. <https://github.com/luuuyi/RefineDet.PyTorch>
9. Pengfei Zhu, Longyin Wen, Xiao Bian, Haibing Ling and Qinghua Hu: Vision Meets Drones: A Challenge (2018)
10. <http://aiskyeye.com/views/getInfo?loc=13>
11. Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun: ShuﬄeNet V2: Practical Guidelines for Eﬃcient CNN Architecture Design (2018)
12. SIMONE BIANCO, REMI CADENE, LUIGI CELONA, AND PAOLO NAPOLETANO: Benchmark Analysis of Representative Deep Neural Network Architectures (2018)
13. <https://github.com/ericsun99/Shufflenet-v2-Pytorch>