Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Домашнее Задание
по дисциплине
«Методы машинного обучения»
на тему

«Обнаружение объектов на основе метода YOLO»

Выполнил:
студент группы ИУ5-21М
Хуан Яовэнь

Москва — 2022 г.

1. Выбор задачи

Обнаружение объектов выбрано в качестве основной задачи домашнего задания. Обнаружение объектов — это задача обнаружения экземпляров определенного класса объектов на изображении. Современные методы можно разделить на два основных типа: одноэтапные методы и двухэтапные методы. Одноэтапные методы отдают предпочтение скорости вывода, и примеры моделей включают YOLO, SSD и RetinaNet. Двухэтапные методы отдают приоритет точности обнаружения, и примеры моделей включают Faster R-CNN, Mask R-CNN и Cascade R-CNN.

Это задание выбирает связанные статьи по методу обнаружения объектов на основе модели YOLO.

Две статьи следующие:

1. You Only Look Once: Unified, Real-Time Object Detection
2. YOLOv4: Optimal Speed and Accuracy of Object Detection

Теоретическая часть в основном описывает связанные принципы YOLO и улучшенную работу YOLOV4.

Практическая часть пытается воспроизвести код обнаружения объектов с помощью YOLOV4.

2. Теоретическая часть

YOLO – это You Only Look Once, то есть YOLO может идентифицировать объекты и их положения на изображении. Традиционная система обнаружения объектов использует метод моделей деформируемых частей (DPM), предлагает целевую область с помощью метода скользящего ящика, а затем использует классификатор для реализации распознавания. В последних методах, подобных R-CNN, используются методы предложения регионов, которые сначала генерируют потенциальные ограничивающие рамки, а затем используют классификатор для определения областей этих ограничивающих рамок. Наконец, постобработка используется для удаления повторяющихся ограничивающих рамок для оптимизации. Этот вид метода сложен в процессе и имеет проблемы с низкой скоростью и трудным обучением.

Поскольку его нужно увидеть только один раз, YOLO называется методом без регионов. По сравнению с методом на основе регионов, YOLO не нужно заранее находить регионы, которые могут иметь цели, также известный как одноэтапный (1 -стадийная) модель. Методы на основе регионов также известны как двухэтапные методы.

Прогноз YOLO основан на всем изображении и выводит сразу всю информацию об обнаруженных объектах, включая категорию и местоположение. Давайте сначала предположим, что изображение, с которым мы имеем дело, является квадратом. Первым шагом YOLO является разделение изображения, он делит изображение на сетки S^2, каждая сетка имеет одинаковый размер, например:

Рис 1. Сетка на картинке

Пусть каждый из S^2 прямоугольников предсказывает ограничивающие прямоугольники B. Этот ограничивающий прямоугольник имеет 5 величин, а

именно положение центра (x, y) объекта, его высоту (h) и ширину (w), и достоверность прогноза (confidence). Каждый блок не только предсказывает B ограничивающих прямоугольников, но и отвечает за предсказание категории объекта в блоке, где категория представлена однократным кодированием (One-hot). Ограничивающий прямоугольник показан ниже:

Рис 2. Ограничивающие прямоугольники

Видно, что одни границы толще, а другие тоньше, это показатели разных уровней уверенности, границы с высокой достоверностью толще, а границы с низкой достоверностью тоньше.

Уверенность представляет долговую расписку и истинность предсказанных блоков. Каждая ячейка сетки предсказывает значение условной вероятности C ($\Pr(Class_i|Object)$). Значение вероятности C представляет вероятность того, что сетка содержит цель, и каждая сетка предсказывает только один тип вероятности. Во время тестирования каждая коробка умножается на вероятность класса и достоверность коробки, чтобы получить конкретную оценку достоверности класса: $\Pr(Class_i|Object) * \Pr(Object) * IOU_{pred}^{truth} = \Pr(Class_i) * IOU_{pred}^{truth}$

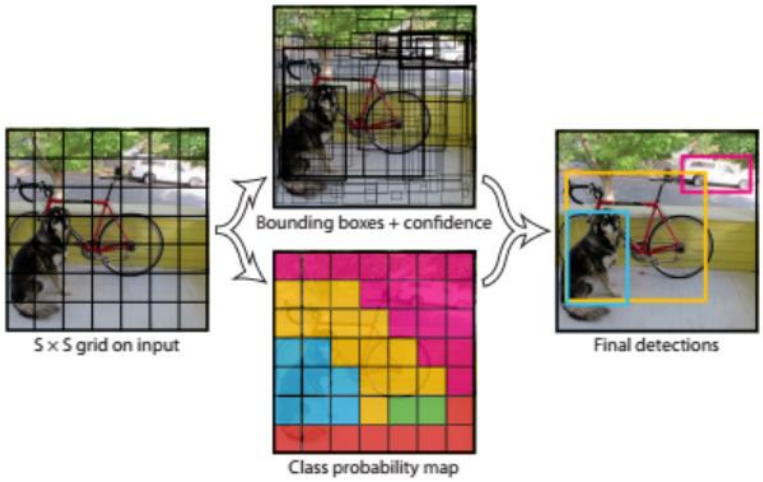

Рис 3. Процесс идентификации

Модель использует структуру сверточной нейронной сети. Начальные сверточные слои извлекают признаки изображения, а полносвязные слои предсказывают выходные вероятности. Структура модели аналогична GoogleNet, как показано на рисунке 4.

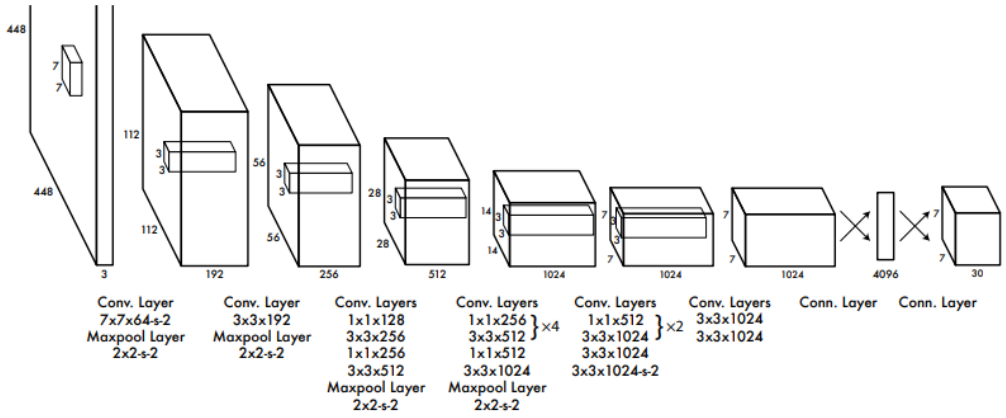

Рис 4. Структура нейронной сети YOLO

Ограничения оригинального YOLO следующие: во-первых, каждая сетка YOLO предсказывает только две коробки, одну категорию. Это приводит к падению точности предсказания модели для соседних объектов. Поэтому YOLO имеет низкую точность распознавания целей с подкладкой (таких как стая птиц). Во-вторых, YOLO учится предсказывать ограничивающие рамки на основе данных, поэтому он не может идентифицировать цели с новыми или

необычными углами. В-третьих, функция потерь YOLO одинаково обрабатывает ошибки малых и больших ограничивающих рамок, что влияет на точность распознавания модели. Потому что для небольших ограничивающих рамок небольшая ошибка оказывает большее влияние.

После нескольких лет технического накопления и итерации версий первоначальный автор YOLO объявил, что прекратит обновление серии YOLO после предложения YOLOv3, позже команда Алексея Бочковского продолжила разработку YOLOv4 на основе YOLOv3. YOLOv4 провел обширные тесты некоторых часто используемых трюков в глубоком обучении и, наконец, выбрал эти полезные приемы: WRC, CSP, CmBN, SAT, активацию Mish, увеличение данных Mosaic, CmBN, регуляризацию DropBlock и потерю CIoU.

YOLOv4 = CSPDarknet53 (магистраль) + дополнительный модуль SPP (шея) + агрегация путей PANet (шейка) + YOLOv3 (голова)

Полная схема структуры нейронной сети выглядит следующим образом:

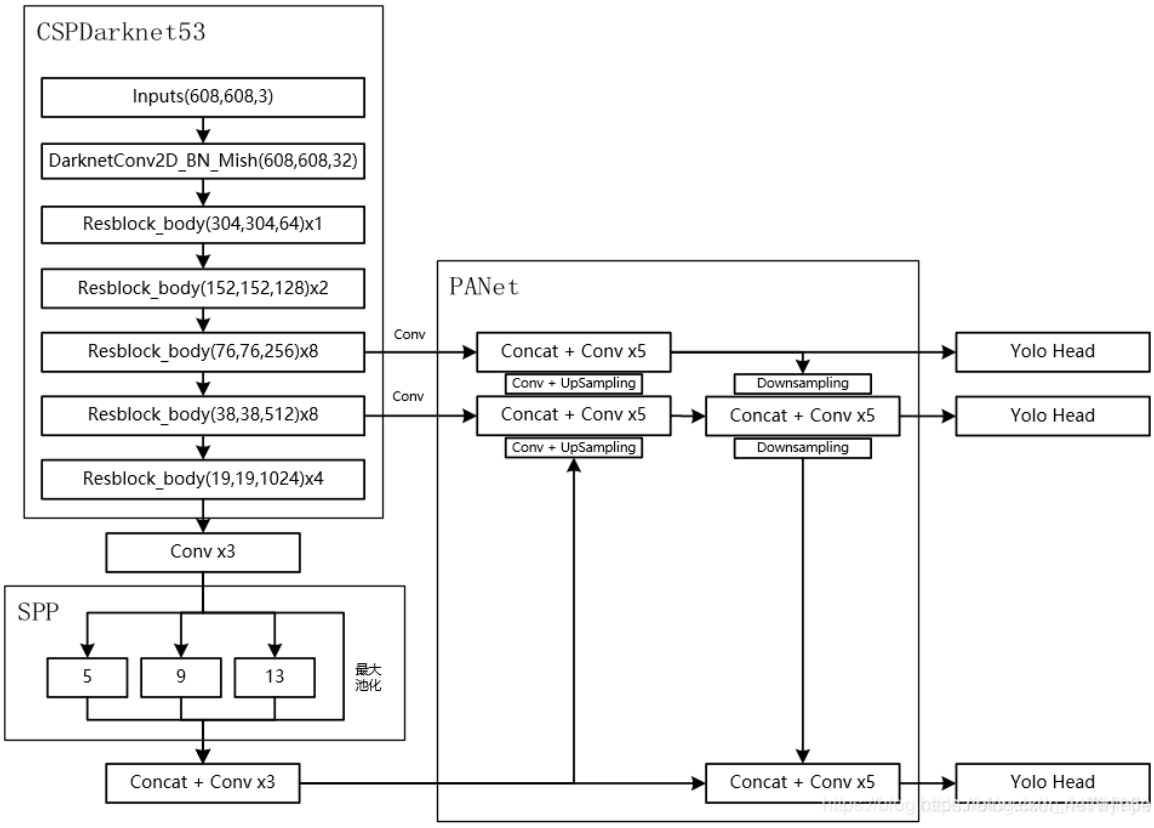

Рис 4. Структура нейронной сети YOLOv4

Поскольку YOLOv4 представляет собой химеру различных методов, в дополнение к сохранению исходной основной идеи обнаружения изображений YOLO, другие были совершенно другими. YOLOv4 — это эффективная и мощная модель обнаружения объектов. Любой может обучить быстрый и высокоточный детектор объектов с помощью графического процессора 1080Ti или 2080Ti. Авторы YOLOv4 улучшили современные алгоритмы, сделав их более эффективными и подходящими для обучения на одном графическом процессоре, например CBN, PAN, SAM и т. д.

В YOLOv4 используются новые функции: WRC, CSP, CmBN, SAT, активацию Mish, увеличение данных Mosaic, CmBN, регуляризацию DropBlock и потерю CIoU, и объединяются некоторые из них для достижения самых современных результатов: 43,5% AP (average precision)(65,7% AP50 (возьмить порог IoU детектора больше 0,5)) для набора данных MS COCO со скоростью в реальном времени ~65 FPS на Tesla V100.

Набор данных MS COCO (Microsoft Common Objects in Context) представляет собой крупномасштабный набор данных для обнаружения объектов, сегментации, обнаружения ключевых точек и подписей. Набор данных состоит из 328 тыс. изображений.

Набор данных имеет аннотации для обнаружение объектов: ограничивающие рамки и маски сегментации для каждого экземпляра с 80 категориями объектов, субтитры: описания изображений на естественном языке (см. Заголовки MS COCO), обнаружение ключевых точек: содержит более 200 000 изображений и 250 000 экземпляров людей, помеченных ключевыми точками (17 возможных ключевых точек, таких как левый глаз, нос, правое бедро, правая лодыжка), сегментация изображения материала - попиксельные маски сегментации с 91 категорией материала, например, трава, стена, небо (см. MS COCO Stuff), паноптический: полная сегментация сцены с 80 категориями вещей (например, человек, велосипед, слон) и подмножеством из 91 категории вещей (трава, небо, дорога), плотная поза: более 39 000 изображений и 56 000 экземпляров людей, помеченных аннотациями DensePose — каждый помеченный человек снабжен аннотацией с идентификатором экземпляра и сопоставлением между пикселями изображения, принадлежащими телу этого человека, и шаблонной 3D-

моделью. Аннотации общедоступны только для обучающих и проверочных изображений.

3. Практическая часть

Практическая часть основана на коде github команды Алексея Бочковского. Запускаем программу в colab.

Чтобы реализовать обучение YOLOv4 на основе платформы darknet, сначала клонируем darknet из репозитория исходного автора на github, настроим Makefile, чтобы включить OPENCV и GPU для даркнета, а затем создаем darknet. Darknet — это относительно легкая среда глубокого обучения с открытым исходным кодом, полностью основанная на C и CUDA. Его основными особенностями являются простота установки, отсутствие зависимостей (можно использовать OpenCV), очень хорошая переносимость и поддержка методов вычислений как на процессоре, так и на графическом процессоре.

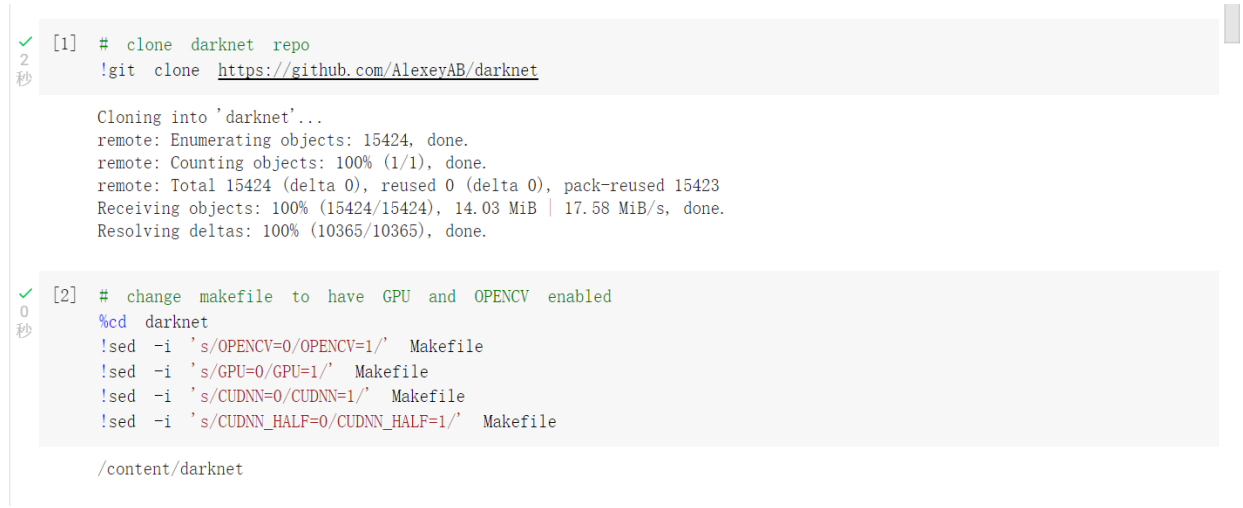

Рис 5. Клонирована библиотека darknet и изменен makefile

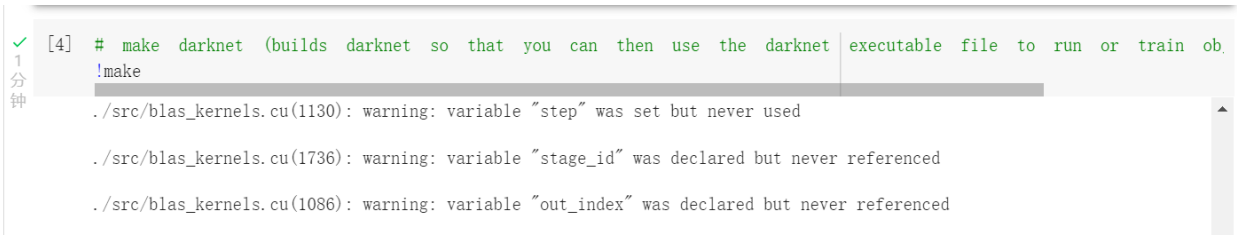

Рис 6. Построена структура darknet

YOLOv4 был обучен на наборе данных COCO, который имеет 80 классов, которые можно предсказать. Мы получим эти предварительно обученные веса, чтобы мы могли запускать YOLOv4 на этих предварительно обученных классах и получать обнаружения.

```
[5]  !wget  https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights

--2022-06-09 21:29:21--  https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/75388965/ba4b6380-889c-11ea-9751-f9
--2022-06-09 21:29:21--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/75388965/ba4b6380-88
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.1
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 257717640 (246M) [application/octet-stream]
Saving to: 'yolov4.weights'

yolov4.weights      100%[===================>] 245.78M   248MB/s    in 1.0s

2022-06-09 21:29:23 (248 MB/s) - 'yolov4.weights' saved [257717640/257717640]
```

Рис 7. Скачать предварительно обученную модель

Создали три вспомогательные функции, которые позволят нам показывать изображение в блокноте Colab после выполнения обнаружений, а также загружать и скачивать изображения в облачную виртуальную машину и из нее.

```
[6]  # define helper functions
     def imShow(path):
         import cv2
         import matplotlib.pyplot as plt
         %matplotlib inline

         image = cv2.imread(path)
         height, width = image.shape[:2]
         resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

         fig = plt.gcf()
         fig.set_size_inches(18, 10)
         plt.axis("off")
         plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
         plt.show()

     # use this to upload files
     def upload():
         from google.colab import files
         uploaded = files.upload()
         for name, data in uploaded.items():
             with open(name, 'wb') as f:
                 f.write(data)
                 print ('saved file', name)
```

```
# use this to download a file
def download(path):
    from google.colab import files
    files.download(path)
```

Рис 8. Определены функции

```
# run darknet detection on test images
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/person.jpg
 92 Shortcut Layer: 89,  wt = 0, wn = 0, outputs:  19 x  19 x 512 0.000 BF
 93 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189 BF
 94 conv    512       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x 512 1.703 BF
 95 Shortcut Layer: 92,  wt = 0, wn = 0, outputs:  19 x  19 x 512 0.000 BF
 96 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189 BF
 97 conv    512       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x 512 1.703 BF
 98 Shortcut Layer: 95,  wt = 0, wn = 0, outputs:  19 x  19 x 512 0.000 BF
 99 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189 BF
100 conv    512       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x 512 1.703 BF
101 Shortcut Layer: 98,  wt = 0, wn = 0, outputs:  19 x  19 x 512 0.000 BF
102 conv    512       1 x 1/ 1     19 x  19 x 512 ->   19 x  19 x 512 0.189 BF
103 route  102 87                              ->   19 x  19 x1024
```

Рис 9. Запуск YOLOv4 на наборе данных COCO

```
imShow('predictions.jpg')
```

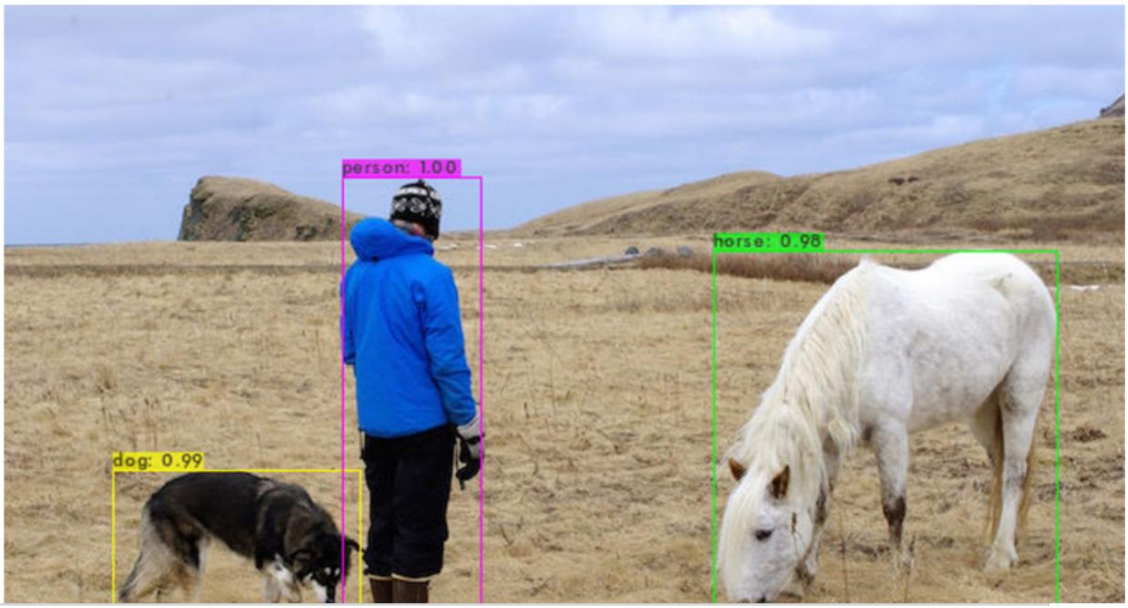

Рис 10. Показать результаты обучения YOLOv4.

После тестирования набора данных COCO на предварительно обученной модели попробовали загрузить изображение для обнаружения объектов.

```
[9]  %cd ..
     upload()
     %cd darknet

     /content
     选择文件 crossroads.jpg
     • crossroads.jpg(image/jpeg) - 109697 bytes, last modified: 2022/6/9 - 100% done
     Saving crossroads.jpg to crossroads.jpg
     saved file crossroads.jpg
     /content/darknet
```

Рис 11. Загрузить файл

Получили следующий результат:

Рис 12. Результат

4. Заключение

Узнали о популярной модели распознавания целей YOLO. Основываясь на работе Алексея Бочковского и др., использовали предварительно обученную модель для распознавания объектов по содержимому изображений.

5. Список использованной литературы

[1] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[2] Bochkovskiy A, Wang C Y, Liao H Y M. Yolov4: Optimal speed and accuracy of object detection[J]. arXiv preprint arXiv:2004.10934, 2020.

[3] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.

[4] AlexeyAB/darknet： YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection （Windows 和 Linux 版本的 Darknet ） (github.com)