

Тренируй пользовательские данные

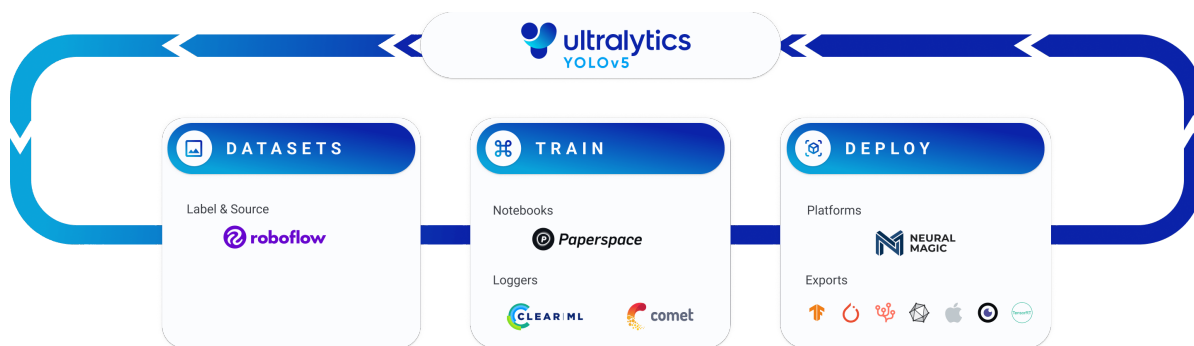
📖 Это руководство объясняет, как обучить свой собственный **набор данных** с помощью YOLOv5 🚀.

Прежде чем начать

Клонируй репо и установи `requirements.txt` в **Python>=3.8.0** в окружении, включая **PyTorch>=1.8**. Модели и наборы данных загружаются автоматически из последнего релиза YOLOv5.

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

Тренируйся на пользовательских данных



Создание пользовательской модели для обнаружения твоих объектов - это итеративный процесс сбора и систематизации изображений, маркировки интересующих тебя объектов, обучения модели, ее развертывания в дикой природе для составления прогнозов, а затем использования этой развернутой модели для сбора примеров крайних случаев для повторения и улучшения.

Лицензирование

Ultralytics предлагает два варианта лицензирования:

- [ЛицензияAGPL-3.0](#) - одобренная [OSI](#) лицензия с открытым исходным кодом, идеально подходящая для студентов и энтузиастов.
- [Лицензия Enterprise](#) для предприятий, которые хотят внедрить наши модели искусственного интеллекта в свои продукты и услуги.

Более подробную информацию ты найдешь на сайте [Ultralytics Лицензирование](#).

YOLOv5 Модели должны быть обучены на меченых данных, чтобы узнать классы объектов в этих данных. Есть два варианта создания набора данных перед началом обучения:

Вариант 1: Создай [Roboflow](#) Набор данных

1.1 Собери изображения

Твоя модель будет учиться на собственном примере. Обучение на изображениях, похожих на те, которые она увидит в естественных условиях, имеет огромное значение. В идеале ты должен собрать множество изображений с той же конфигурацией (камера, ракурс, освещение и т.д.), с которой ты в итоге будешь разворачивать свой проект.

Если это невозможно, ты можешь начать с [публичного набора данных](#), чтобы обучить свою начальную модель, а затем в [процессе вычисления выбирать изображения из дикой природы](#), чтобы итеративно улучшать набор данных и модель.


1.2 Создание ярлыков

Когда ты соберешь изображения, тебе нужно будет аннотировать интересующие тебя объекты, чтобы создать базовую истину, на которой будет обучаться твоя модель.

3

Preprocessing

Decrease training time and increase performance by applying image transformations to all images in this dataset.

Auto-Orient	Edit	×
Resize Stretch to 640×640	Edit	×
 Add Preprocessing Step		

Continue

- **Auto-Orient** - чтобы убрать EXIF-ориентацию из твоих изображений.
- **Изменить размер (Stretch)** - до квадратного входного размера твоей модели (640x640 - это YOLOv5 по умолчанию).

Создав версию, ты получишь моментальный снимок своего набора данных, так что ты всегда сможешь вернуться и сравнить с ним будущие тренировки модели, даже если впоследствии добавишь больше изображений или изменишь ее конфигурацию.

Let us train your model and get results within 24 hours

Export

Format

YOLO v5 PyTorch

TXT annotations and YAML config used with [YOLOv5](#).

☐ download zip to computer ☒ show download code

Cancel Continue

Экспорт в YOLOv5 Pytorch формат, затем скопируй сниппет в свой обучающий скрипт или блокнот, чтобы загрузить свой набор данных.


```
1: bicycle
2: car
# ...
77: teddy bear
78: hair drier
79: toothbrush
```

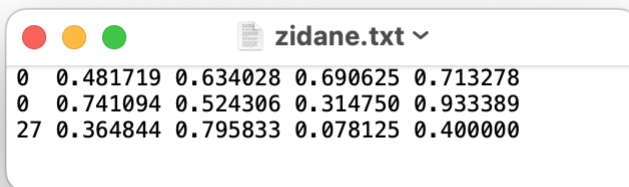
2.2 Создание ярлыков

После использования инструмента для создания аннотаций к изображениям, экспортируй свои метки в **YOLO Формат**, с одним *.txt файл на изображение (если в изображении нет объектов, то нет и *.txt файл обязателен). Сайт *.txt Спецификации файлов таковы:

- Один ряд на объект
- Каждый ряд - это `class x_center y_center width height` формат.
- Координаты коробки должны быть в **нормализованный xwh** формат (от 0 до 1). Если твои поля в пикселях, раздели `x_center` и `width` по ширине изображения, и `y_center` и `height` по высоте изображения.
- Номера классов имеют нулевую индексацию (начинаются с 0).



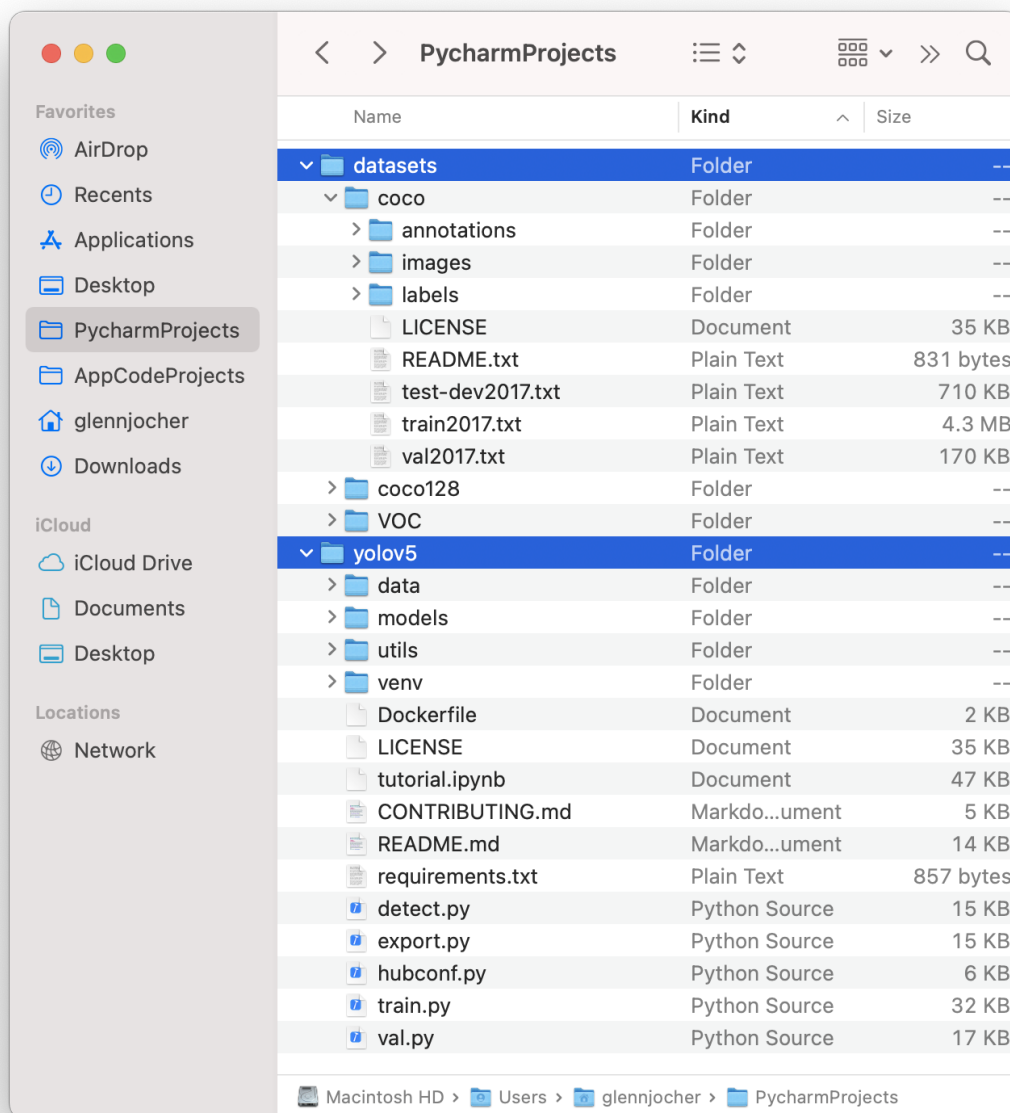
Файл с метками, соответствующий изображению выше, содержит 2 человека (класс 0) и
ничья (класс 27):



2.3 Организуй каталоги





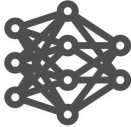
Организуй изображения и метки для поездов и валов в соответствии с приведенным ниже примером. YOLOv5 предполагает. `/coco128` находится внутри `/datasets` каталог рядом с `/yolov5` каталог. **YOLOv5 Автоматически определяет местонахождение этикеток для каждого изображения** заменив последний экземпляр `/images/` в каждом пути изображения с `/labels/`. Например:

```
../datasets/coco128/images/im0.jpg # image
../datasets/coco128/labels/im0.txt # label
```



3. Выберите модель

Выбери предварительно обученную модель, с которой начнешь обучение. Здесь мы выбрали **YOLOv5s**, вторую по размеру и самую быструю из доступных моделей. Полное сравнение всех моделей смотри в нашей [таблице](#) README.

				
Nano YOLOv5n	Small YOLOv5s	Medium YOLOv5m	Large YOLOv5l	XLarge YOLOv5x
4 MB _{FP16} 6.3 ms _{V100} 28.4 mAP _{COCO}	14 MB _{FP16} 6.4 ms _{V100} 37.2 mAP _{COCO}	41 MB _{FP16} 8.2 ms _{V100} 45.2 mAP _{COCO}	89 MB _{FP16} 10.1 ms _{V100} 48.8 mAP _{COCO}	166 MB _{FP16} 12.1 ms _{V100} 50.7 mAP _{COCO}

4. Поезд

Обучи модель YOLOv5s на COCO128, указав набор данных, размер партии, размер изображения и либо предварительно обученную модель `--weights yolov5s.pt` (рекомендуется), или случайно инициализированный `--weights '' --cfg yolov5s.yaml` (не рекомендуется). Предварительно обученные веса автоматически загружаются из [Последний выпуск YOLOv5](#).

```
python train.py --img 640 --epochs 3 --data coco128.yaml --weights yolov5s.pt
```



Наконечник



Добавь `--cache ram` или `--cache disk` для ускорения тренировок (требует значительных ресурсов оперативной памяти/диска).



Наконечник



Всегда тренируйся на локальном наборе данных. Смонтированные или сетевые диски вроде Google Drive будут работать очень медленно.

Все результаты тренировок сохраняются в `runs/train/` с увеличивающимися директориями запуска, то есть `runs/train/exp2`, `runs/train/exp3` и так далее. Подробнее об этом читай в разделе "Обучение" нашего обучающего блокнота.



Open in Colab



Open in Kaggle

5. Визуализируй

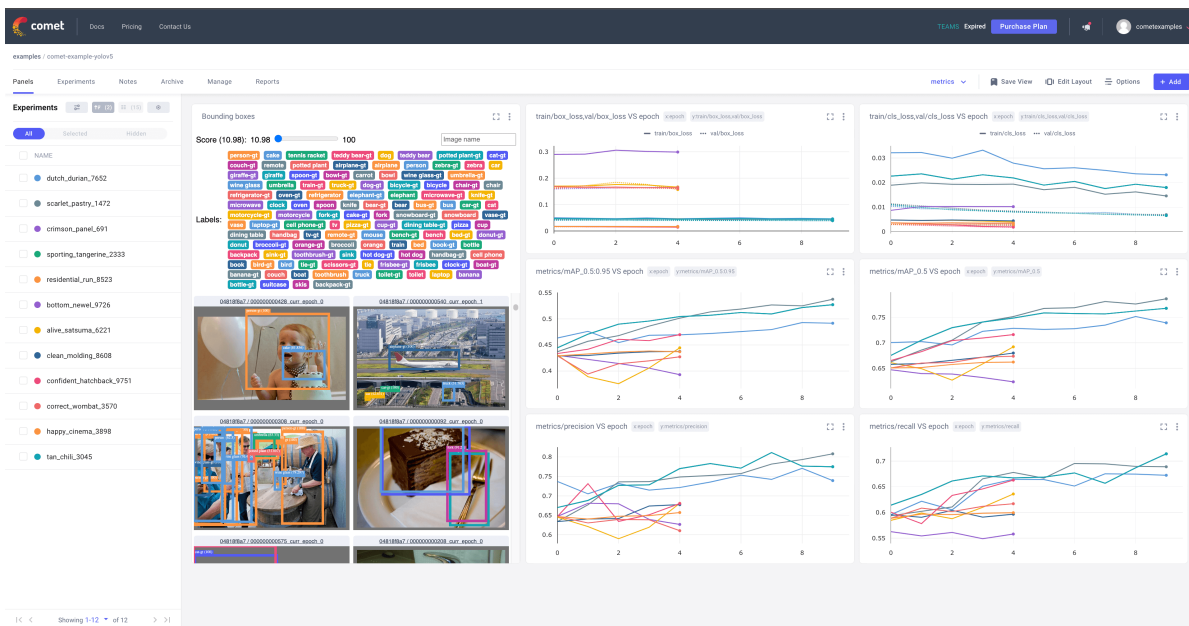
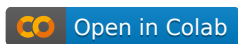
Comet Логирование и визуализация 🌟 НОВИНКА

[Comet](#) теперь полностью интегрирован с YOLOv5. Отслеживай и визуализируй показатели модели в реальном времени, сохраняй свои гиперпараметры, наборы данных и контрольные точки модели, а также визуализируй предсказания модели с помощью [Comet Custom Panels](#)! Comet позволяет тебе никогда не терять контроль над своей работой и легко делиться результатами и сотрудничать в командах любого размера!

Приступить к работе очень просто:

```
pip install comet_ml # 1. install
export COMET_API_KEY=<Your API Key> # 2. paste API key
python train.py --img 640 --epochs 3 --data coco128.yaml --weights yolov5s.pt #
3. train
```

Чтобы узнать больше обо всех поддерживаемых функциях Comet для этой интеграции, ознакомься с [Comet Самоучитель](#). Если ты хочешь узнать больше о Comet, заходи на наш сайт. [документация](#). Начни с того, что попробуй блокнот Comet Colab Notebook:



ClearML Логирование и автоматизация 🌟 НОВИНКА

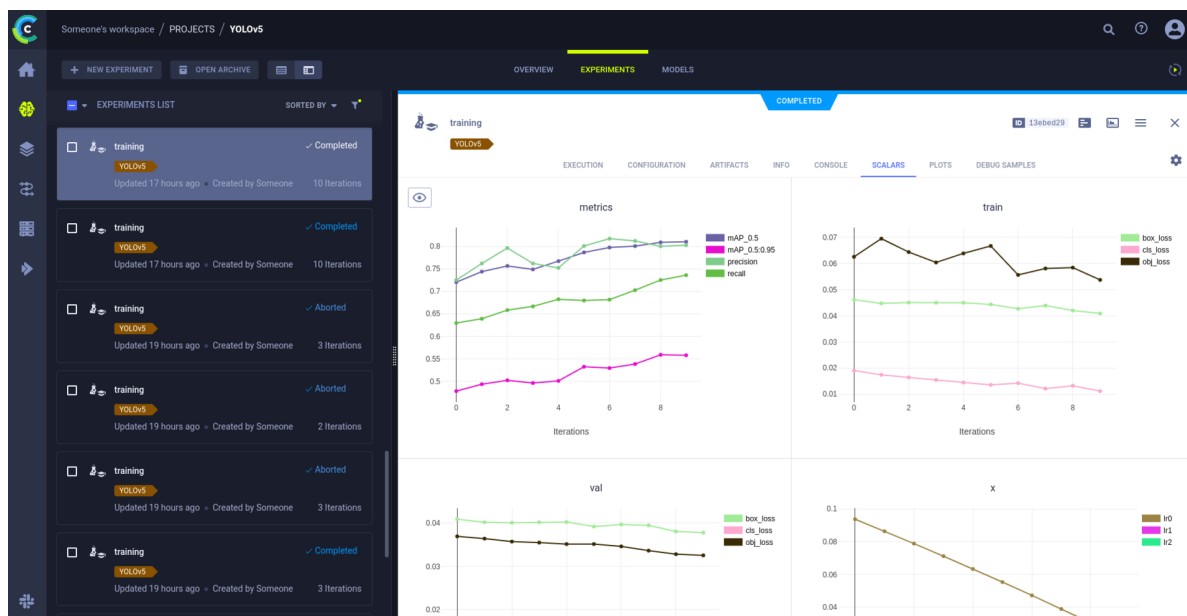


[ClearML](#) полностью интегрирован в YOLOv5, чтобы отслеживать ход твоих экспериментов, управлять версиями наборов данных и даже удаленно выполнять тренировочные прогоны. Чтобы включить ClearML:

- `pip install clearml`
- запусти `clearml-init` чтобы подключиться к серверу ClearML

Ты получишь все отличные функции, ожидаемые от менеджера экспериментов: обновления в реальном времени, загрузку моделей, сравнение экспериментов и т. д., но ClearML также отслеживает незафиксированные изменения и, например, установленные пакеты. Благодаря этому ClearML Tasks (именно так мы называем эксперименты) также воспроизводимы на разных машинах! С помощью всего 1 дополнительной строки мы можем запланировать YOLOv5 тренировочную задачу в очередь на выполнение любым количеством ClearML -агентов (рабочих).

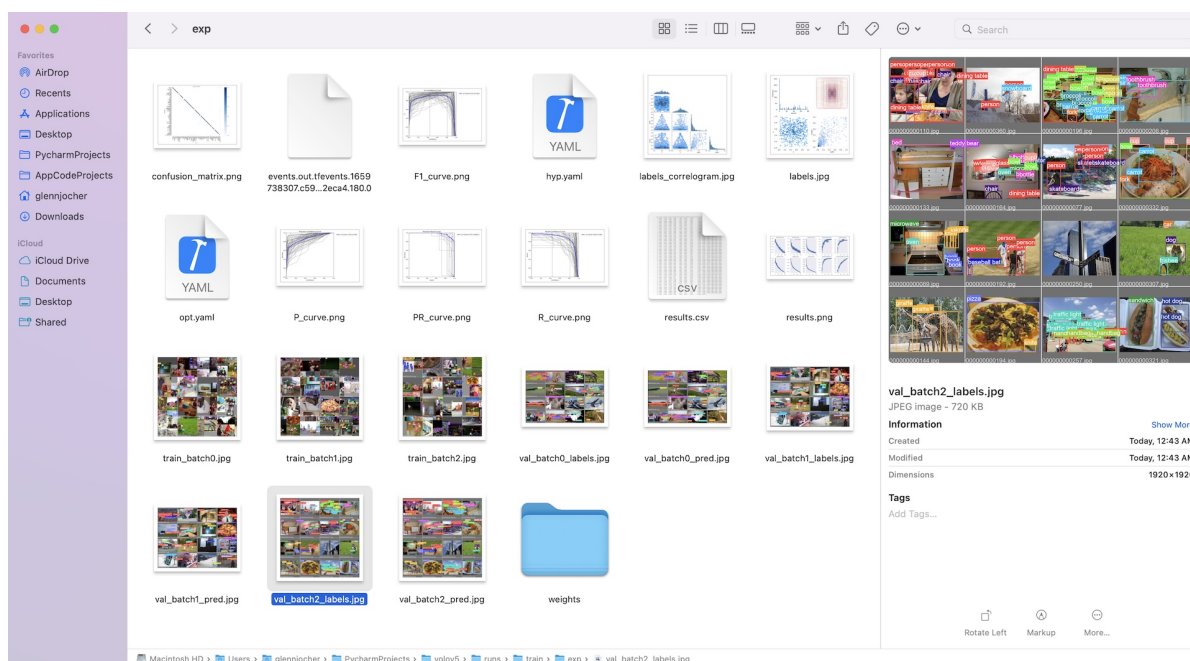
Ты можешь использовать ClearML Data для версионирования своего набора данных, а затем передать его на YOLOv5, просто используя его уникальный ID. Это поможет тебе следить за своими данными без лишних хлопот. Изучи [учебник ClearML](#), чтобы узнать подробности!



Локальное протоколирование

Результаты тренировок автоматически регистрируются с помощью [Tensorboard](#) и [CSV](#) логиры, чтобы `runs/train` При этом для каждой новой тренировки создается новый каталог экспериментов. `runs/train/exp2`, `runs/train/exp3`, и т. д.

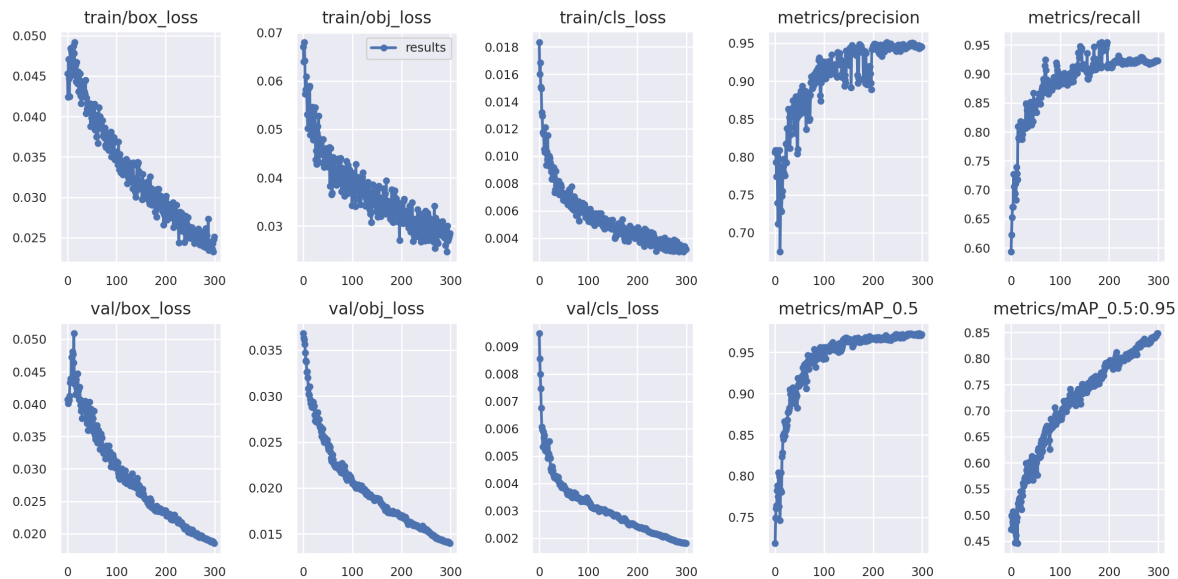
Этот каталог содержит статистику по train и val, мозаики, метки, предсказания и дополненные мозаики, а также метрики и графики, включая кривые precision-recall (PR) и матрицы путаницы.



Файл с результатами `results.csv` обновляется после каждой эпохи, а затем строится как `results.png` (ниже) после завершения тренировки. Ты также можешь построить график любого `results.csv` файл вручную:

```
from utils.plots import plot_results

plot_results('path/to/results.csv') # plot 'results.csv' as 'results.png'
```



Следующие шаги

Когда твоя модель обучена, ты можешь использовать свой лучший контрольный пункт `best.pt` чтобы:

- Запусти [CLI](#) или [Python](#) умозаключение на новых изображениях и видео
- [Проверь](#) точность на тренировочном, валидном и тестовом сплитах
- [Экспортируй](#) в форматы TensorFlow, Keras, ONNX, TFLite, TF.js, CoreML и TensorRT .
- [Развивай](#) гиперпараметры, чтобы улучшить производительность
- [Улучши](#) свою модель, взяв образцы изображений из реального мира и добавив их в свой набор данных

Поддерживаемые среды

Ultralytics Он предоставляет ряд готовых к использованию окружений, в каждом из которых предустановлены такие необходимые зависимости, как [CUDA](#), [cuDNN](#), [Python](#), и [PyTorch](#), чтобы запустить твои проекты.

- **Бесплатные ноутбуки с графическим процессором:** 

 [Open in Colab](#)  [Open in Kaggle](#)

- **Google Cloud:** [Руководство по быстрому запуску GCP](#)
- **Amazon:** [Руководство по быстрому запуску AWS](#)

- **Azure:** [Руководство по быстрому запуску AzureML](#)

- **Docker:** [Руководство по быстрому запуску Docker](#)  docker pulls 326k

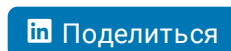
Статус проекта



Этот значок означает, что все тесты непрерывной интеграции (CI) [YOLOv5 GitHub Actions](#) успешно пройдены. Эти CI-тесты тщательно проверяют функциональность и производительность YOLOv5 по различным ключевым аспектам: [обучение](#), [валидация](#), [вывод](#), [экспорт](#) и [бенчмарки](#). Они обеспечивают стабильную и надежную работу на macOS, Windows и Ubuntu, причем тесты проводятся каждые 24 часа и при каждом новом коммите.

Создано 2023-11-12, Обновлено 2024-01-21

Авторы: [glenn-jocher](#) (11)



Комментарии

0 reactions



0 comments

