PyTorch Hub

📚 В этом руководстве объясняется, как загрузить YOLOv5 🚀 с сайта PyTorch Hub на https://pytorch.org/hub/ ultralytics_yolov5.

Прежде чем начать

Установите файл requirements.txt в **Python>=3.8.0** в окружении, включающем **PyTorch>=1.8**. Модели и наборы данных загружаются автоматически из последнегорелиза YOLOv5 .

```
pip install -r
https://raw.githubusercontent.com/ultralytics/yolov5/master/requirements.txt
```

💡 ProTip: Клонирование https://github.com/ultralytics / yolov5 **не** обязательно 😃.

Загрузи YOLOv5 с PyTorch Hub

Простой пример

Этот пример загружает предварительно обученную модель YOLOv5s с сайта PyTorch Hub как `model` и передаёт изображение для вывода. `'yolov5s'` это самая легкая и быстрая модель YOLOv5 . Подробнее обо всех доступных моделях читай в разделе README.

```
import torch

# Model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')

# Image
im = 'https://ultralytics.com/images/zidane.jpg'

# Inference
results = model(im)

results.pandas().xyxy[0]
#      xmin    ymin    xmax   ymax  confidence  class    name
```

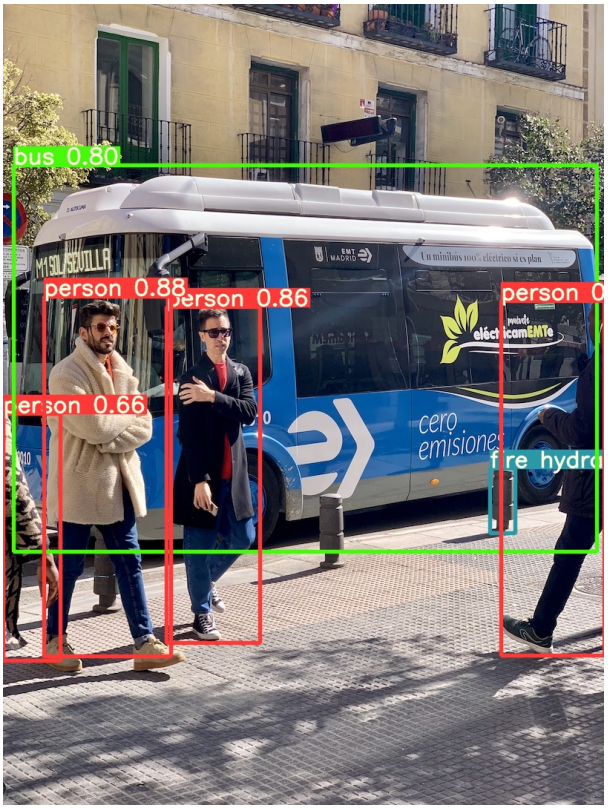
RU

```
# 0  749.50   43.50  1148.0  704.5    0.874023      0  person
# 1  433.50  433.50   517.5  714.5    0.687988     27     tie
# 2  114.75  195.75  1095.0  708.0    0.624512      0  person
# 3  986.00  304.00  1028.0  420.0    0.286865     27     tie
```

Подробный пример

Этот пример показывает. **пакетное умозаключение** с **PIL** и **OpenCV** Источники изображения. `results` может быть **Напечатанный** на консоль, **Сохранил** на `runs/hub`, **показал** для проверки на поддерживаемых средах и возвращается в виде **тензоры** или **панды** датафреймы.

```python
import cv2
import torch
from PIL import Image

# Model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')

# Images
for f in 'zidane.jpg', 'bus.jpg':
    torch.hub.download_url_to_file('https://ultralytics.com/images/' + f, f)  # download 2 images
im1 = Image.open('zidane.jpg')  # PIL image
im2 = cv2.imread('bus.jpg')[..., ::-1]  # OpenCV image (BGR to RGB)

# Inference
results = model([im1, im2], size=640)  # batch of images

# Results
results.print()
results.save()  # or .show()

results.xyxy[0]  # im1 predictions (tensor)
results.pandas().xyxy[0]  # im1 predictions (pandas)
```



Обо всех вариантах вывода см. YOLOv5 `AutoShape()` вперед метод.

Настройки умозаключений

YOLOv5 Модели содержат различные атрибуты вывода, такие как **порог уверенности**, **порог IoU** и т. д., которые можно задавать:

RU

```
model.conf = 0.25  # NMS confidence threshold
iou = 0.45  # NMS IoU threshold
agnostic = False  # NMS class-agnostic
multi_label = False  # NMS multiple labels per box
classes = None  # (optional list) filter by class, i.e. = [0, 15, 16] for COCO
persons, cats and dogs
max_det = 1000  # maximum number of detections per image
amp = False  # Automatic Mixed Precision (AMP) inference

results = model(im, size=320)  # custom inference size
```

Устройство

Модели после создания можно переносить на любое устройство:

```
model.cpu()  # CPU
model.cuda()  # GPU
model.to(device)  # i.e. device=torch.device(0)
```

Модели также могут быть созданы непосредственно на любом `device`:

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', device='cpu')  # load on
CPU
```

💡 ProTip: Входные изображения автоматически переносятся на нужное модельное устройство перед умозаключением.

Тихие выходы

Модели можно загружать бесшумно с помощью `_verbose=False`:

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', _verbose=False)  # load
silently
```

Входные каналы

Чтобы загрузить предварительно обученную модель YOLOv5s с 4 входными каналами, а не 3, как по умолчанию:

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', channels=4)
```

В этом случае модель будет состоять из предварительно обученных весов, **за исключением** самого первого входного слоя, который уже не будет иметь ту же форму, что и предварительно обученный входной слой. Входной слой по-прежнему будет инициализирован случайными весами.

Количество классов

Чтобы загрузить предварительно обученную модель YOLOv5s с 10 выходными классами, а не 80, как по умолчанию:

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', classes=10)
```

В этом случае модель будет состоять из предварительно обученных весов, **за исключением** выходных слоев, которые уже не будут иметь ту же форму, что и предварительно обученные выходные слои. Выходные слои по-прежнему будут инициализированы случайными весами.

Принудительная перезагрузка

Если у тебя возникнут проблемы с выполнением вышеописанных действий, установи `force_reload=True` может помочь удаление существующего кэша и принудительная загрузка последней версии YOLOv5 с сайта PyTorch Hub .

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', force_reload=True)  # force reload
```

Умозаключение по скриншоту

Чтобы запустить умозаключения на экране твоего рабочего стола:

```
import torch
from PIL import ImageGrab

# Model
```

RU

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')

# Image
im = ImageGrab.grab()  # take a screenshot

# Inference
results = model(im)
```

Мультипроцессорные выводы

YOLOv5 Модели могут быть загружены на несколько GPU параллельно с потоковым выводом:

```
import torch
import threading


def run(model, im):
    results = model(im)
    results.save()


# Models
model0 = torch.hub.load('ultralytics/yolov5', 'yolov5s', device=0)
model1 = torch.hub.load('ultralytics/yolov5', 'yolov5s', device=1)

# Inference
threading.Thread(target=run, args=[model0,
'https://ultralytics.com/images/zidane.jpg'], daemon=True).start()
threading.Thread(target=run, args=[model1,
'https://ultralytics.com/images/bus.jpg'], daemon=True).start()
```

Тренировка

Чтобы загрузить модель YOLOv5 для обучения, а не для вывода, установи `autoshape=False`. Чтобы загрузить модель со случайно инициализированными весами (для обучения с нуля), используй `pretrained=False`. В этом случае ты должен предоставить свой собственный сценарий тренировки. В качестве альтернативы смотри наш YOLOv5 Учебник по работе с пользовательскими данными для обучения модели.

```
import torch

model = torch.hub.load('ultralytics/yolov5', 'yolov5s', autoshape=False)  load
```

RU

```
pretrained
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', autoshape=False,
pretrained=False)  # load scratch
```

Результаты в формате Base64

Для использования с API-сервисами. Подробности смотри на сайте https://github.com/ultralytics/yolov5/pull/2291 и в примере Flask REST API.

```
results = model(im)  # inference

results.ims  # array of original images (as np array) passed to model for
inference
results.render()  # updates results.ims with boxes and labels
for im in results.ims:
    buffered = BytesIO()
    im_base64 = Image.fromarray(im)
    im_base64.save(buffered, format="JPEG")
    print(base64.b64encode(buffered.getvalue()).decode('utf-8'))  # base64
encoded image with results
```

Обрезанные результаты

Результаты могут быть возвращены и сохранены в виде культур обнаружения:

```
results = model(im)  # inference
crops = results.crop(save=True)  # cropped detections dictionary
```

Результаты работы панд

Результаты могут быть возвращены в виде Pandas DataFrames:

```
results = model(im)  # inference
results.pandas().xyxy[0]  # Pandas DataFrame
```

Pandas Output (нажмите, чтобы развернуть)

RU

Отсортированные результаты

Результаты можно сортировать по столбцам, то есть отсортировать определение цифр номерного знака слева направо (ось x):

```
results = model(im)  # inference
results.pandas().xyxy[0].sort_values('xmin')  # sorted left-right
```

Обрезанные результаты

Результаты могут быть возвращены и сохранены в виде культур обнаружения:

```
results = model(im)  # inference
crops = results.crop(save=True)  # cropped detections dictionary
```

Результаты в формате JSON

Результаты могут быть возвращены в формате JSON после преобразования в `.pandas()` фреймы данных, используя `.to_json()` Метод. Формат JSON можно изменить, используя `orient` Аргумент. Смотри pandas `.to_json()` документация подробнее.

```
results = model(ims)  # inference
results.pandas().xyxy[0].to_json(orient="records")  # JSON img1 predictions
```

Выходные данные в формате JSON (нажмите, чтобы развернуть)

Пользовательские модели

Этот пример загружает пользовательский 20-класс VOC-обученная модель YOLOv5s `'best.pt'` с PyTorch Hub .

```
import torch

model = torch.hub.load('ultralytics/yolov5', 'custom', path='path/to/best.pt')  #
local model
```

RU

```
model = torch.hub.load('path/to/yolov5', 'custom', path='path/to/best.pt',
source='local')  # local repo
```

TensorRT, ONNX и OpenVINO Модели

PyTorch Hub поддерживает вывод данных в большинстве форматов экспорта YOLOv5 , включая пользовательские обученные модели. Подробности экспорта моделей см. в учебнике TFLite, ONNX, CoreML, TensorRT Export.

💡 ProTip: **TensorRT** может быть в 2-5 раз быстрее, чем PyTorch в **бенчмарки GPU** 💡 ProTip: **ONNX** и **OpenVINO** может быть в 2-3 раза быстрее, чем PyTorch в бенчмарках для CPU. **бенчмарки CPU**

```
import torch

model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5s.pt')  #
PyTorch
model = torch.hub.load('ultralytics/yolov5', 'custom',
path='yolov5s.torchscript')  # TorchScript
model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5s.onnx')  #
ONNX
model = torch.hub.load('ultralytics/yolov5', 'custom',
path='yolov5s_openvino_model/')  # OpenVINO
model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5s.engine')  #
TensorRT
model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5s.mlmodel')  #
CoreML (macOS-only)
model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5s.tflite')  #
TFLite
model = torch.hub.load('ultralytics/yolov5', 'custom',
path='yolov5s_paddle_model/')  # PaddlePaddle
```

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**:   
- **Google Cloud**: Руководство по быстрому запуску GCP

RU

- **Amazon**: Руководство по быстрому запуску AWS
- **Azure**: Руководство по быстрому запуску AzureML
- **Docker**: Руководство по быстрому запуску Docker 

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 2023-12-03
Авторы: glenn-jocher (3)

Комментарии