Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерный наук и технологий
Высшая школа программной инженерии

КУРСОВАЯ РАБОТА

«Разработка веб-сервера для развертывания моделей глубокого обучения на базе Flask и Docker»
по дисциплине «Сети и телекоммуникации»

Выполнила
студентка гр. 3530904/90102 Ли Ицзя

Руководитель Шакуро П. Е.

Санкт-Петербург
2023

Оглавление

Постановка задачи

С развитием технологии глубокого обучения все больше и больше приложений нуждаются в развертывании моделей глубокого обучения на веб-серверах. Однако в этом процессе есть определенные технические пороги и проблемы. Как быстро, стабильно и эффективно развертывать модели глубокого обучения на веб-серверах — актуальная проблема, требующая решения.

Flask — это популярный веб-фреймворк Python, созданный на основе Werkzeug и Jinja2. Он известен тем, что он прост, легок в освоении, легковесен и предлагает широкие возможности расширения и настройки. Flask также поддерживает запросы и ответы RESTful и может быстро создавать интерфейсы API. В этом проекте мы используем Flask в качестве среды веб-разработки для создания серверных приложений.

Docker — это облегченная контейнерная технология, позволяющая упаковать приложение в независимый переносимый контейнер, чтобы приложение могло работать в любой среде. Контейнерная технология Docker имеет следующие преимущества: 1. контейнерные приложения можно легко развернуть в различных операционных системах, виртуальных машинах или платформах облачных вычислений; 2. контейнерные приложения имеют независимые файловые системы и сетевые стеки, избегая конфликтов и помех между различными приложениями; 3. Контейнерные приложения можно легко расширять и управлять ими. В этом проекте мы используем Docker для упаковки модели глубокого обучения в контейнер и развертывания контейнера на сервере.

Цели Работы

1. Упаковать обученную модель в контейнер Docker и запустите контейнер на сервере.
2. Использовать фреймворк Flask для разработки интерфейса веб-API, через который клиент может загружать файлы изображений и получать результаты обнаружения цели.
3. На стороне сервера библиотека OpenCV используется для обработки изображения, загруженного клиентом, а развернутая модель YOLOv5 используется для обнаружения цели.
4. Вернуть результат обнаружения цели клиенту.

Проведение работы

Аппаратные среды, использованные для разработки, показаны в Таблице 1:

Таблица 1

Предмет	Технические характеристики	Комментарий
Чип	AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz	
Паияти	16.0 GB	ПО использует до 2.1 GB

Операционная системы	Windows 10 Professional	

Программные среды, использованные в разработке, показаны а Таблице 2:

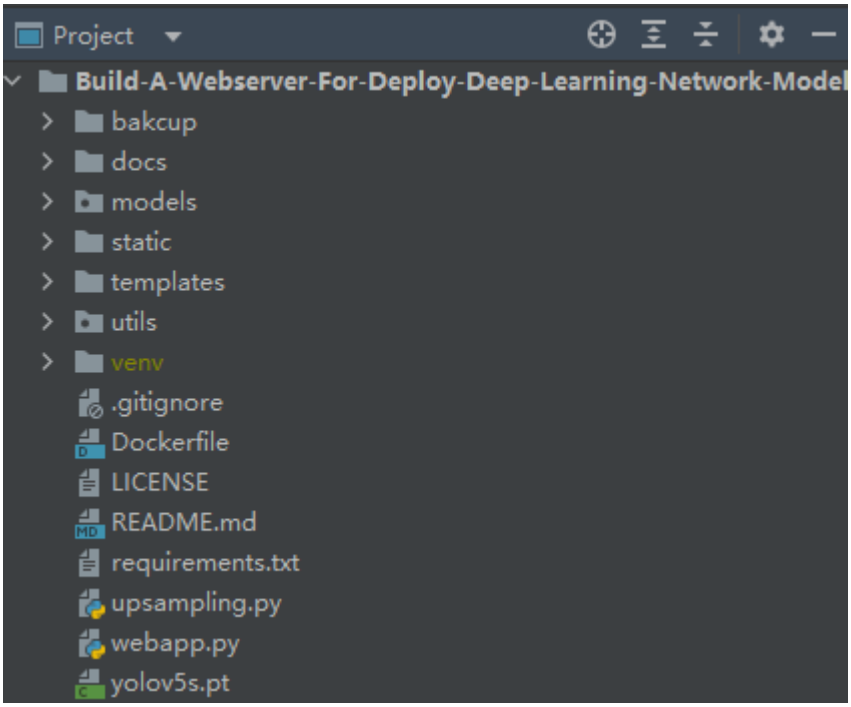
Таблица 2

Предмет	Версия	Комментарий
Python	3.8.16	Данная или выше
PyCharm	2022.1.2 (Educational Edition)	Данная или выше
PyTorch	2.0.0	Данная или выше

Локальное Развертывание

Структура Проекта

Общая структура проекта показана на рисунке ниже



models: Хранить программы, связанные с построением модели

utils: Хранить чертежи, загрузку данных и другие сопутствующие инструменты

static: Храненить статические файлы

templates: HTML-файл главной страницы

webapp.py: вступительная программа

yolov5s.pt: Предварительно обученная модель YOLO

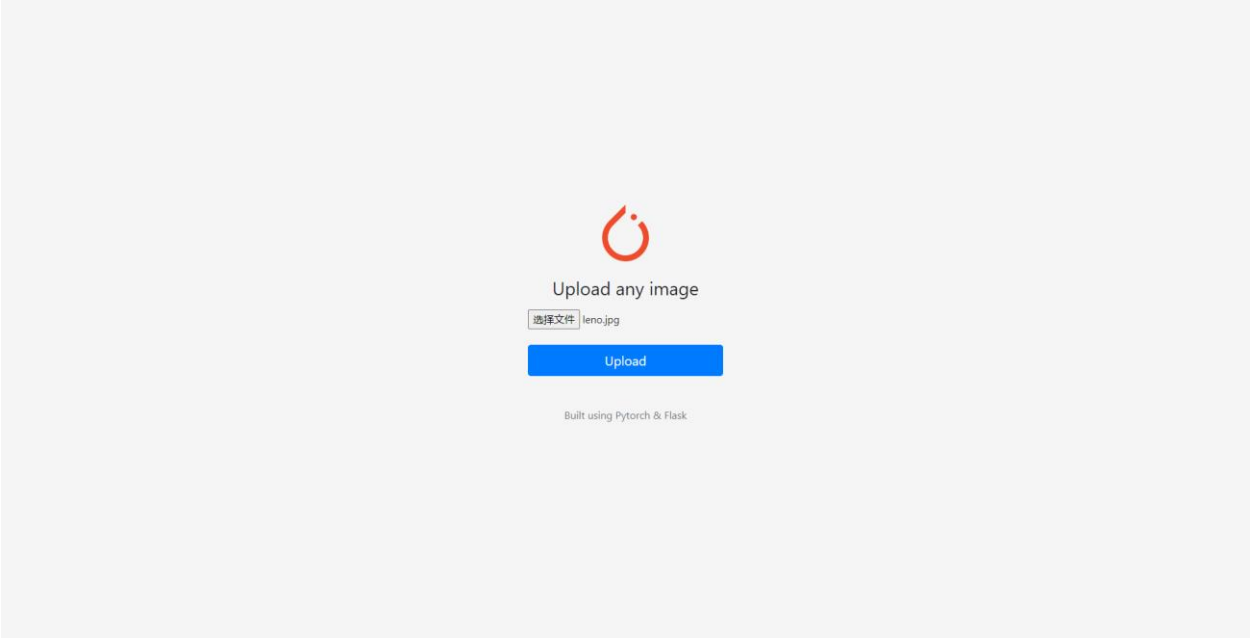
requiments.txt: Зависимости Python, необходимые для запуска проекта

upsamling.py: В файле upsampling.py текущей версии pytorch есть избыточность параметра recompute_scale_factor, вот измененный исходный код.

Dockerfile: Файл, используемый для создания образов Docker

Запустить Проект

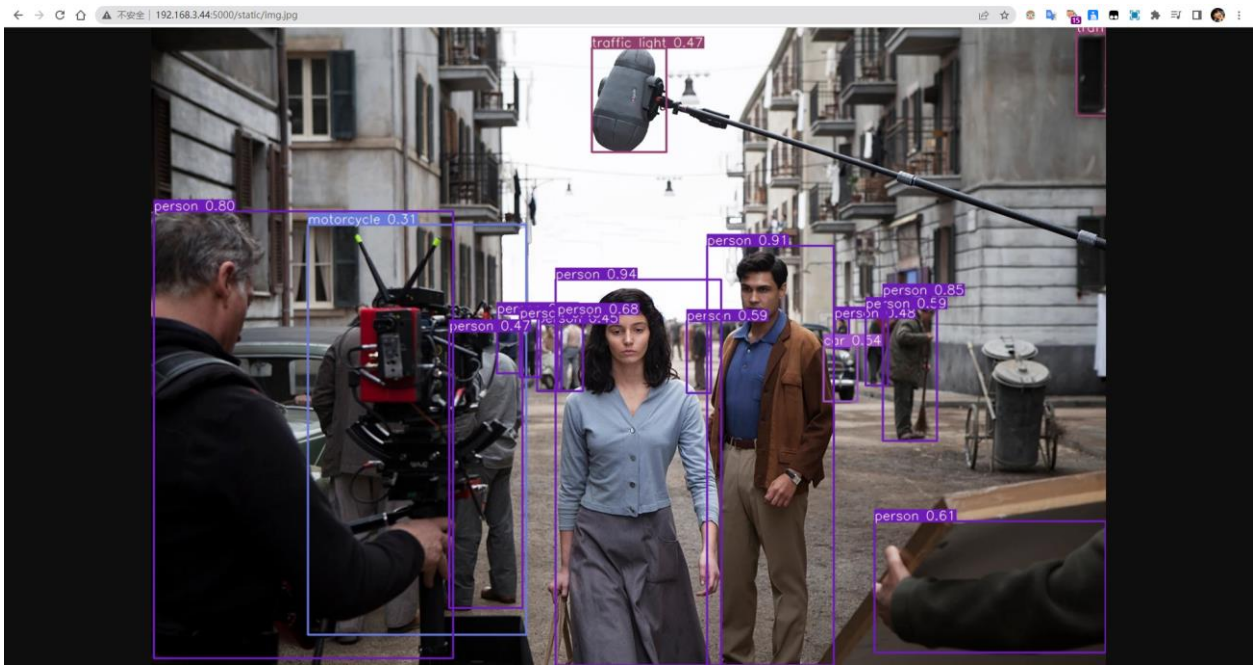
Запустите `python webapp.py` в терминале, подождите немного, вы можете посетить http://127.0.0.1:5000 и увидите отрендеренную страницу.



Выберите файл на главной странице и загрузите его, чтобы вернуть результат прогнозирования модели. Предсказанные изображения будут сохранены в статической папке.

Файл теста:

Результат:



Облачное Развертывание

Создание requiments.txt

Использовала библиотеку pipreqs. Он может выполнять некоторую фильтрацию и выводить только библиотеки и версии, используемые в проекте.

```
coremltools==6.2
flask==2.2.2
```

```
matplotlib==3.7.1
numpy==1.23.5
onnx==1.13.1
opencv_python==4.7.0.72
pafy==0.5.5
pandas==1.5.3
Pillow==9.4.0
PyYAML==6.0
requests==2.28.2
scipy==1.10.1
seaborn==0.12.2
thop==0.1.1.post2209072238
torch==2.0.0
torchvision==0.15.1
tqdm==4.65.0
```

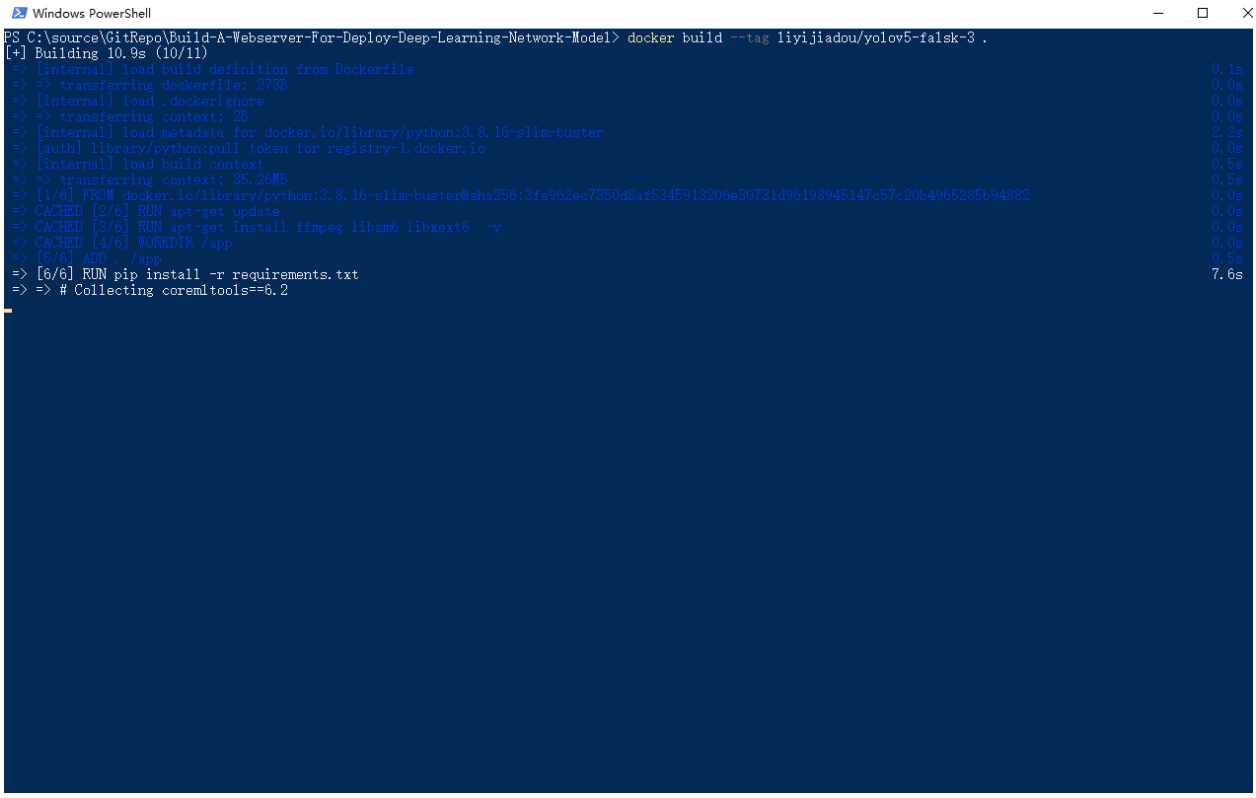
Написать Dockerfile

Содержимое построенного DockerFile выглядит следующим образом:

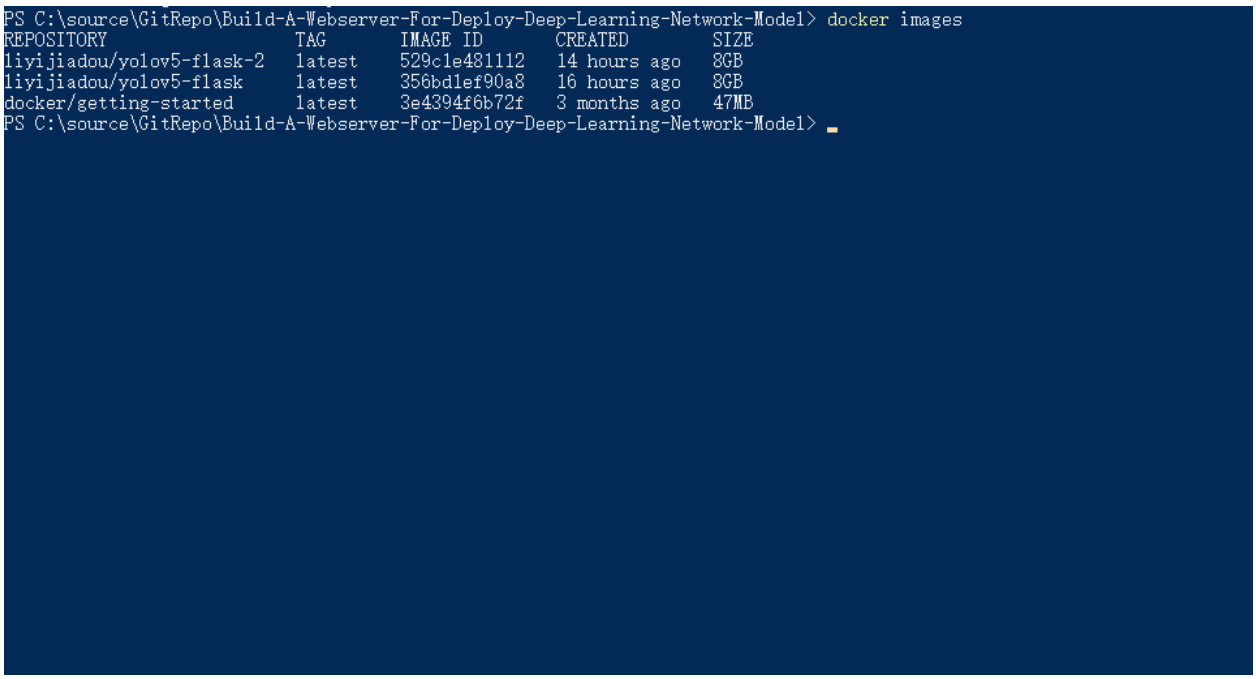
```
FROM python:3.8.16-slim-buster
MAINTAINER liyijiadou

RUN apt-get update
RUN apt-get install ffmpeg libsm6 libxext6  -y

WORKDIR /app
ADD . /app
RUN pip install -r requirements.txt

EXPOSE 5000

CMD ["python", "webapp.py"]
```

Сделать image с Помощью Dockerfile

```
docker build --tag liyijiadou/yolov5-flask .
```



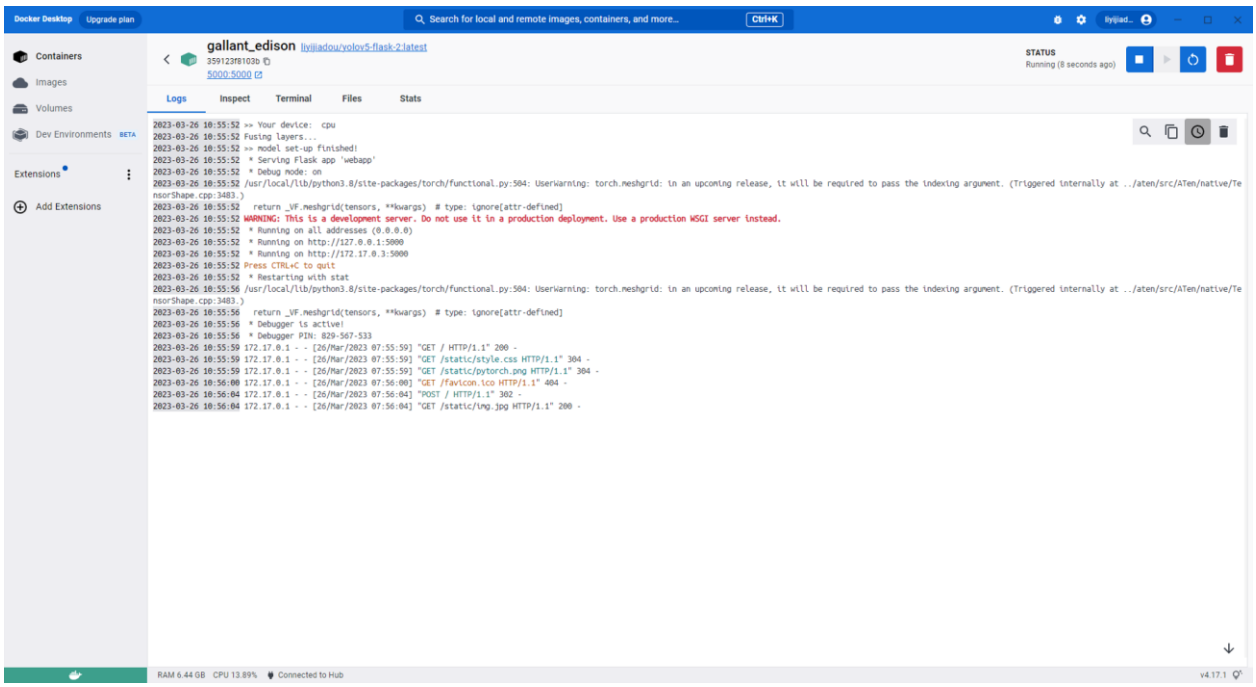
После этого, получим docker image:



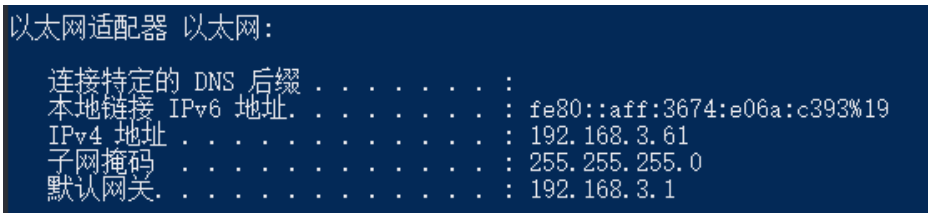
Создать Docker Container и Запус

```
docker run -p 5000:5000 liyijiadou/yolov5-flask
```

После этого, получим docker container:

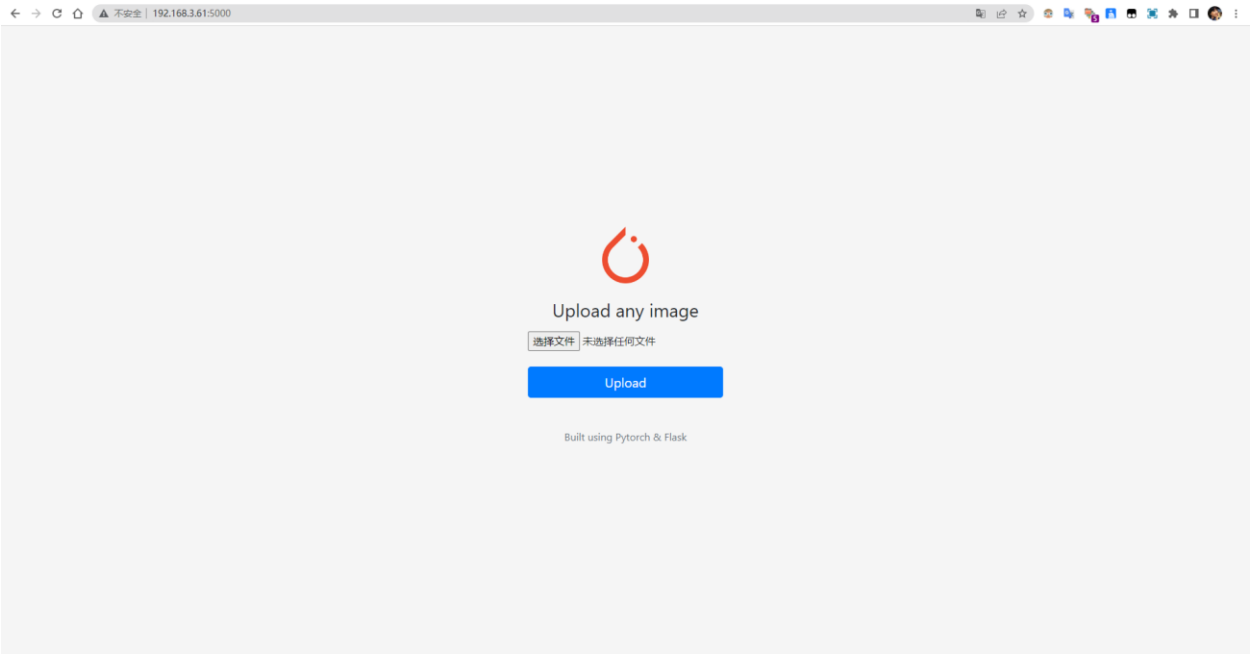


IP адрес: 192.168.3.1



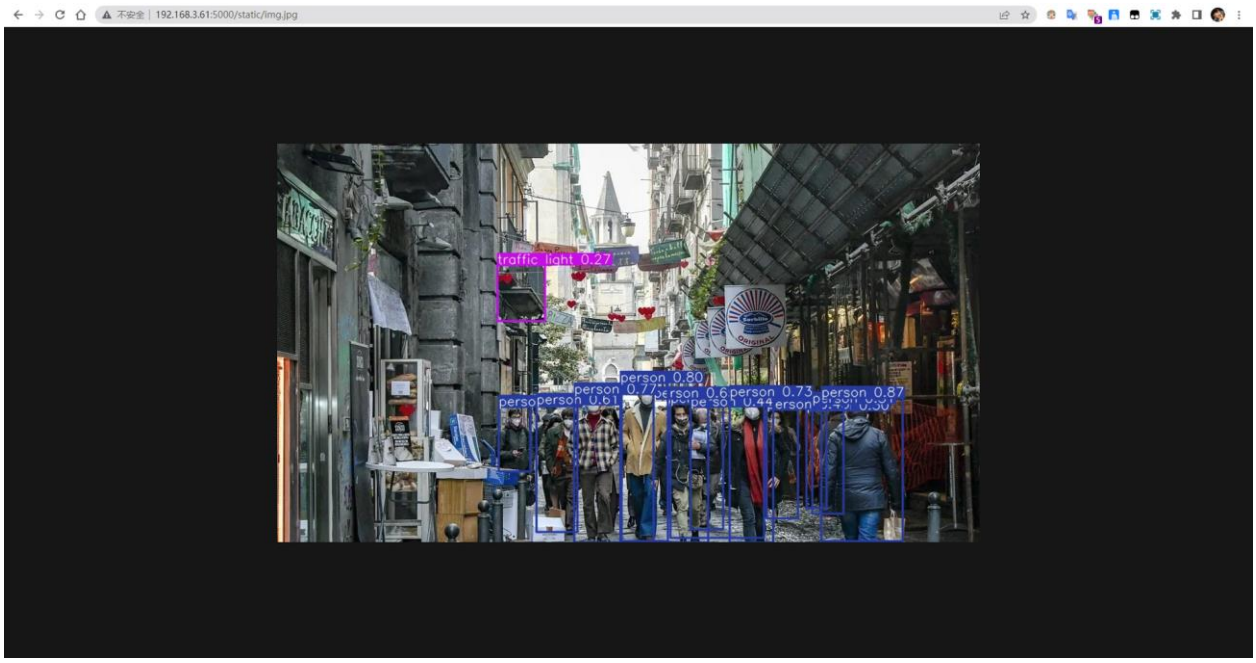
Результат

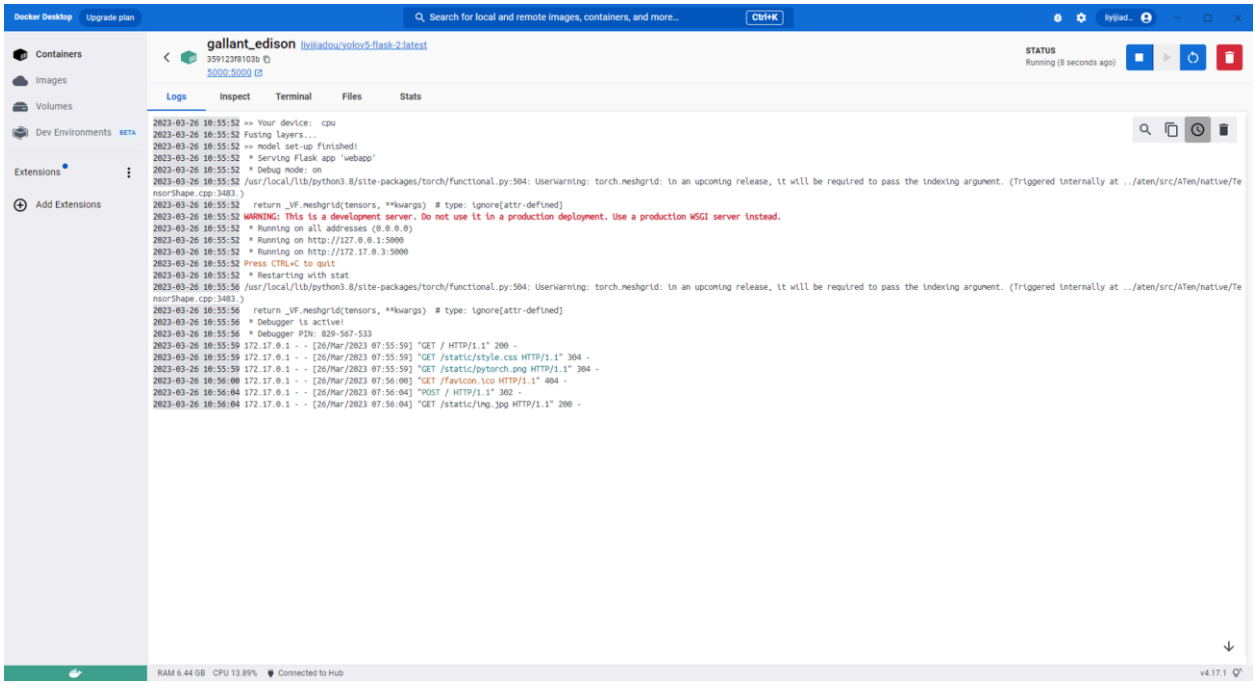
Запустите контейнер, загрузите картинку, и вы увидите, что результат рассуждений представлен правильно!



Файл теста:

Результат:





Вывод

После вышеуказанных шагов мы успешно разработали веб-сервер на основе Flask и Docker и развернули модель глубокого обучения. Сервер может получать файлы изображений, загруженные пользователями, и вызывать модель глубокого обучения для их прогнозирования. В тесте я использовала обученную модель обнаружения объектов для обнаружения загружаемых изображений на стороне сервера. Результаты тестирования показывают, что сервер обладает стабильностью, высокой эффективностью и масштабируемостью.

Приложение

Docker image для этого проекта
https://hub.docker.com/repository/docker/liyijiadou/yolov5-flask/general

Github
https://github.com/liyijiadou2020/Build-A-Webserver-For-Deploy-Deep-Learning-Network-Model.git