

Санкт-Петербургский политехнический университет Петра Великого

Институт металлургии, машиностроения и транспорта

Высшая школа автоматизации и робототехники

Курсовая работа

Дисциплина: Объектно-ориентированное программирование

Тема: Модуль аугментации датасетов

Выполнил студент гр. 3331506/10401

Макаров Г.В.

Преподаватель

Ананьевский М.С.

«___»_____ 2024 г.

Санкт-Петербург

2024

Оглавление

Цель.....	3
Введение.....	4
Ход работы.....	5
Заключение.....	10
Список литературы.....	11

Цель

Необходимо разработать модуль аугментации датасетов для детекции объектов по ограничивающим рамкам на языке программирования Python.

Инструмент должен будет включать следующий функционал:

1. Сканирование датасета в поисках изображений и аннотаций к ним, занесение найденных данных в словарь.
2. Выбор аугментаций и соответствующих параметров, определяющих характер аугментации.
3. Аугментация всех изображений и аннотаций для каждого из запрошенных пользователем субсетов – тренировочного, валидационного и/или тестового.
4. Сохранение аугментированного датасета.

Для реализации модуля будут использованы только библиотеки NumPy, Pillow для уменьшения количества зависимостей и общего веса.

Ссылка на репозиторий реализованного проекта -

<https://gitverse.ru/makarov/augmentor>. Исходный код во вложении.

Введение

Аугментация датасетов для задач детекции объектов — это процесс увеличения размера существующего набора данных путем добавления новых модифицированных изображений и соответствующих им аннотаций. Это опциональный шаг при обучении моделей машинного обучения для задач компьютерного зрения (классификация, детекция, сегментация и т.д.).

Аугментация, прежде всего, решает проблему малого количества изначальных данных. В реальных бизнес-задачах инженеры машинного обучения часто сталкиваются с весьма ограниченным датасетом, расширить который за счет интернет-ресурсов не представляет возможным. За счёт аугментации можно многократно увеличить объём данных.

Еще одна проблема, которую решает аугментация — это улучшение обобщающей способности моделей. Это особенно важно при работе с изображениями, где даже небольшие изменения, такие как зашумление или поворот, могут существенно изменить внешний вид объекта. Так, например, аугментация с поворотом изображений на произвольный угол полезна для датасетов с аэрофотоснимками, где аннотации объекта с разной ориентацией позволят идентифицировать его независимо от положения камеры.

Ход работы

Существует несколько форматов хранения датасетов для задач детекции объектов. Самые распространенные из них – COCO, CVAT и YOLO. В рамках задачи будет реализован последний формат, как самый популярный и активный в сообществе компьютерного зрения. Информацию о нём можно найти по ссылке <https://docs.ultralytics.com/ru/datasets/detect/>. Главное отличие от более старых форматов, вроде COCO состоит в том, что координаты – x , y , w , h – рассчитываются относительно размеров всего изображения. Так, например, 1.0 значит, что координата занимает всё изображение вдоль определенной оси, а 0.5 – только половину.

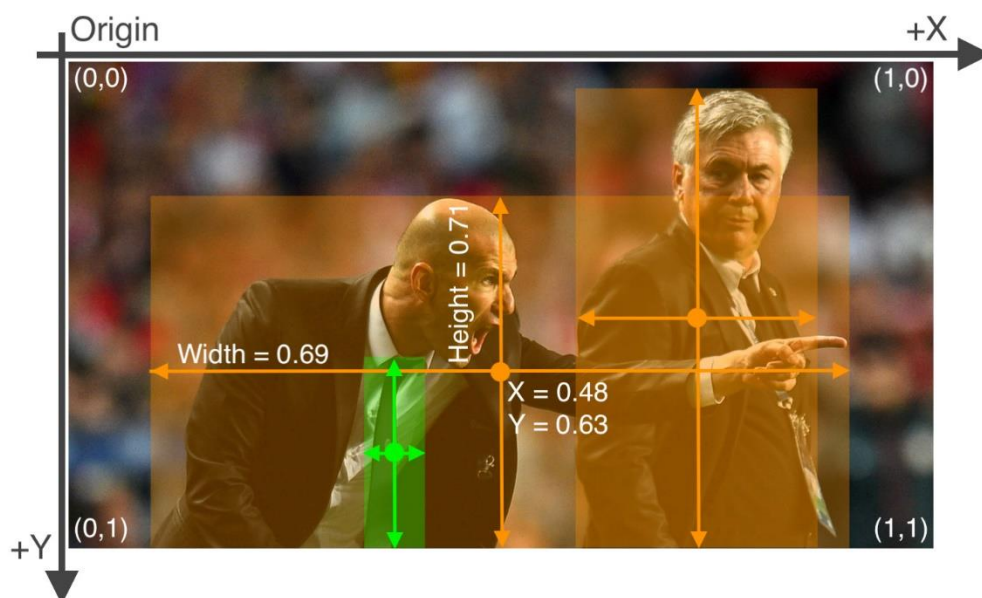


Рисунок 1 – Формат датасетов YOLO

Чтобы работать с таким форматом, а в дальнейшем легко встраивать любой другой, реализуем интерфейс AugmentorBase. Тогда функционал сканирования датасетов формата YOLO будет включён в дочерний класс, который будет наследовать методы AugmentorBase. В родительском классе потребуются общие инструменты: *логгер, аугментация, чтение и запись изображений*. Соответственно, остальные функции будут абстрактными и их нужно будет определять в каждом наследнике: *сканирование датасетов, чтение и запись аннотаций*. Реализация интерфейса предоставлена в файле

augmentor/core/base.py, а дочерний класс вынесен в отдельный файл **augmentor/core/formats/yolo.py**, согласно структуре проекта (см. вложение).

Чтобы пользователь не работал с каждым отдельным дочерним классом под интересные ему форматы имеет смысл сделать единый класс – Augmentor – который будет создавать нужный класс в зависимости от запрошенного пользователем формата. Такой подход называется фабричным шаблоном проектирования (Factory Pattern), и реализуется несколькими способами. В модуле для аугментации это было решено сделать через одноименный класс с переопределенным магическим методом `__new__`. Реализация этого паттерна прописана в **augmentor/__init__.py**.

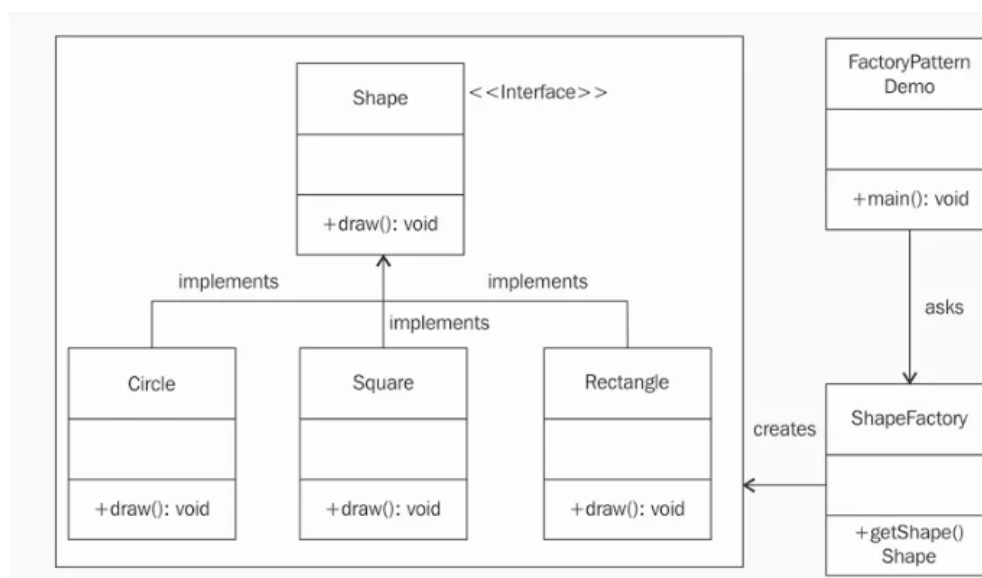


Рисунок 2 – Factory Pattern на примере фигур

Центральной частью модуля станут сами аугментации – методы, которые будут принимать на вход изображения и аннотации. Аугментации делятся на геометрические и фотометрические. *Геометрические аугментации* влияют на геометрию изображения и на связанные с этим позиции ограничивающих рамок. *Фотометрические аугментации* меняют свойства изображений, вроде яркости, оставляя геометрию и позицию ограничивающих рамок без изменений. Разницу можно лицезреть, сравнивая оригинальное изображений (рис. 3) и аугментированные (рис. 4 и 5).



Рисунок 3 - Оригинальное изображение

Геометрические аугментации предоставлены во вложении в файле **augmentor/augmentations/geometric.py**. Среди реализованных:

1. Поворот изображений на 90, 180 или 270 градусов (или на -270, -180 и -90 соответственно).
2. Вращение изображений вокруг оси x , y или вокруг обеих.
3. Изменение размера изображений на величину от 0.0 до 2.0 – уменьшение или увеличение соответственно.
4. Урезание изображения по координатам x , y , w , h .

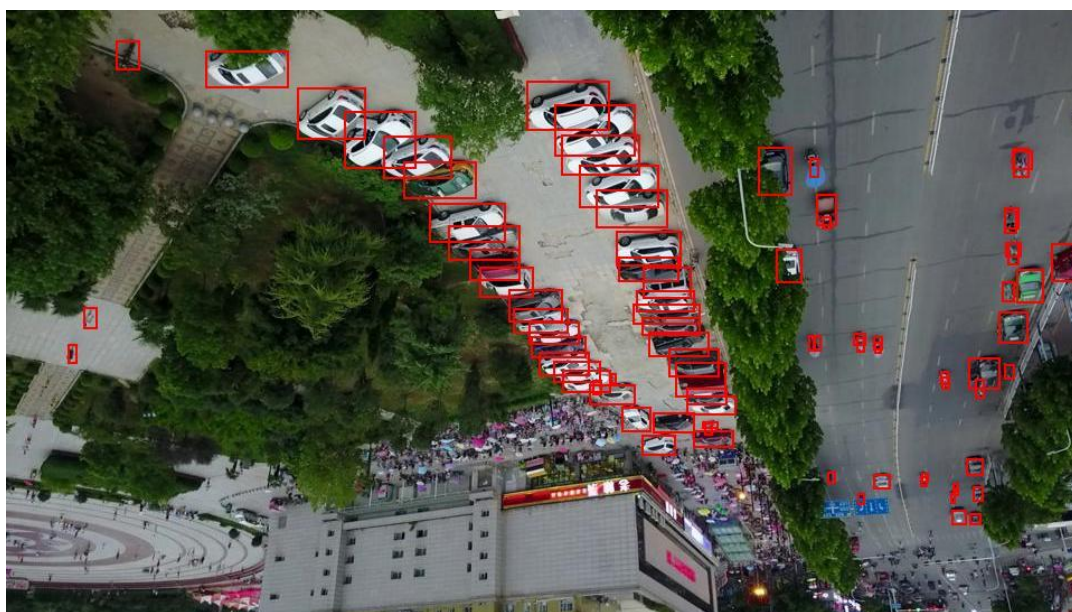


Рисунок 4 – Геометрическая аугментация (вращение вокруг осей x и y)

Фотометрические аугментации предоставлены во вложении в файле **augmentor/augmentations/photometric.py**. Среди реализованных:

1. Изменение насыщенности изображений на фактор от 0.0 до 2.0 (с помощью библиотеки Pillow).
2. Изменение контраста изображений на фактор от 0.0 до 2.0 (с помощью библиотеки Pillow).
3. Изменение яркости изображений на фактор от 0.0 до 2.0 (с помощью библиотеки Pillow).
4. Размытие Гаусса с заданным значением стандартного гауссовского распределения от 0.0 до 2.0.
5. Шум Гаусса с заданным значением стандартного гауссовского распределения от 0.0 до 2.0.



Рисунок 5 – Фотометрическая аугментация (шум Гаусса)

На этом основная часть модуля готова, и остаётся только оформить его в виде устанавливаемого пакета. Для этого используем соответствующую структуру проекта, частично описанную выше, и **pyproject.toml**, который позволит устанавливать и импортировать наш модуль в дальнейшем. Окончательный результат можно увидеть во вложении или в репозитории.

Также был оформлен **README.md** и написан docstring для всех публичных функций и методов модуля.

Чтобы установить модуль для его использования в других проектах, нужно будет клонировать репозиторий:

```
git clone https://gitverse.ru/sc/makarov/augmentor.git
```

И установить все его зависимости:

```
pip3 install augmentor/.
```

Чтобы использовать модуль после его установки необходимо создать объекта класса Augmentor с заданными форматом, который через реализованный фабричный паттерн позволит получить только необходимый для этого формата функционал. Опционально можно включить дебаггер и указать путь для логгера, куда будут записываться данные о ходе аугментации. Наконец, через созданный объект класса можно вызвать функцию *augment* – в качестве параметра передать пайплайн аугментации, который представляет из себя череду строчных названий и параметров (например, “*rotate*”, 90 для вращения на 90 градусов). В обязательном порядке указываются пути для загрузки и сохранения датасета. Опционально указываются интересующие сабсеты для аугментации (изначально сканируются все). Суммируя, простой пример использования модуля выглядит следующим образом:

```
from augmentor import Augmentor
```

```
augmentor = Augmentor(fmt="YOLO", debug=True, log_path="/home/user/Documents/log.txt")
```

```
augmentor.augment("blur", 1.2,  
                  "resize", 0.8,  
                  load_path="/home/user/Documents/VisDrone",  
                  save_path="/home/user/Documents/VisDroneAugmented",  
                  train=True, val=True, test=True)
```

Заключение

В результате работы был успешно написан модуль, который позволяет решать задачи, связанные с аугментацией датасетов для задач детекции объектов. Инструмент был протестирован и показал сравнительно высокую скорость работы. Он может быть использован специалистами компьютерного зрения для расширения малых датасетов в качестве stand-alone проекта. При желании, модуль может быть легко интегрирован в существующие системы и приложения.

В перспективе для модуля требуется написать поддержку форматов датасетов COCO и CVAT for image. Также планируется написать CLI и, возможно, минималистичный GUI, но потребуется реализовать кроссплатформенность инструмента – поддержка как Linux, так и Windows. В будущем может быть интегрирована поддержка ориентированных аннотаций (повернутых на определенный угол ограничивающих рамок).

Список литературы

1. Буслаев, А. (2019). Albumentations: быстрая библиотека аугментации изображений в Python. [Онлайн]. Доступно: <https://albumentations.readthedocs.io>
2. Lin, T. Y., et al. (2014). Microsoft COCO: обзор базы данных большого масштаба для обнаружения объектов, классификации и других задач. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2015 (стр. 1355-1363).
3. Shorten, C., и Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), 60.
4. Howard, J., et al. (2019). Fastai: A Layered API for Deep Learning. Information, 11(8).
5. Источников, А. (2020). Image augmentation library for machine learning with Keras and TensorFlow in Python. [Онлайн]. Доступно: <https://github.com/aleju/imgaug>
6. Redmon, J., и Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. [Онлайн]. Доступно: <https://arxiv.org/abs/1612.08242>