Welcome to Automatic Object and Action Detection's documentation!

Документация по детектированию объектов и действий

Требования 1

Nvidia GPU (GTX 650 or newer)
CUDA Toolkit v11.2
CuDNN 8.1.0
CUDA Toolkit - Anaconda Python 3.7
OS - Windows, Linux

Требования 2

Python - 3.7
TensorFlow >= 2.5, <=2.8

Установка необходимых модулей

Клонирование ветки репозитория с моделями Tensorflow

```
git clone https://github.com/tensorflow/models.git
```

Репозиторий

Ссылка на репозиторий с кодом и зависимостями - https://github.com/elina-chertova/tensorflow-2-object-detection

```
pip3 install -r requirements.txt
```

Установка Object Detection

```
cd models/research/
protoc object_detection/protos/*.proto --python_out=.
cp object_detection/packages/tf2/setup.py .
python -m pip install .
```

Тестирование установки Object Detection

```
python models/research/object_detection/builders/model_builder_tf2_test.py
```

Начало обучения

Внутри проекта необходимо создать папку, в которой будут храниться изображения и их аннотации.

```
mkdir your_folder
```

Разметку можно сделать с помощью - https://github.com/tzutalin/labelImg

Затем необходимо загрузить размеченные данные в your_folder.

1. Запустить функции из prepare_data.py для предобработки датасета
2. Перед запуском auto_object_detection.py можно выбрать свои параметры класса.
   Модель можно выбрать из предложенных по следующей ссылке - https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Готовая модель

1. Файлы, сгенерированные кодом (pipeline.config, data.csv, label_map.pbtxt, train_data.record, test_data.record)
   pipeline.config — содержит все параметры для обучения конкретной модели (/your_folder)
   label_map.pbtxt — файл с описанием классов (/annotations)

data.csv — файл с данными об изображениях и их разметках (/your_folder)

train_data.record, test_data.record — файл с данными из data.csv в формате, необходимом tensorflow для обучения (/annotations)

2. Все чекпоинты модели находятся в папке /your_folder/output
3. Готовая модель находится в папке saved_model, your_folder/output/frozen/saved_model

Структура

Класс prepare_data

class **PrepareData**(*image_folder='images_data', train_set_percent=0.8*) [исходный код]

__init__(*image_folder='images_data', train_set_percent=0.8*) [исходный код]

create_test_and_train_folder() [исходный код]

Создает тестовую и обучающую папки

Результат:

divide_test_and_train() [исходный код]

Создает папки test и train и рандомно раскидывает по ним изображения и соответствующие xml файлы

move_all_images_to_images_directory() [исходный код]

Переносит все изображения и аннотации в /all_images_data

remove_files_without_pair() [исходный код]

Удаляет изображения, у которых нет аннотаций, и аннотации, у которых нет изображения

remove_unreadable_images() [исходный код]

Удаляет поврежденные изображения формата jpg/jpeg и соотвествующие xml

Класс auto_object_detection

class **ObjectDetection**(*folder_dataset_name='images_data', model_number=20, batch_size=12, num_steps=200000, use_custom_num_steps=False*) [исходный код]

__init__(*folder_dataset_name='images_data', model_number=20, batch_size=12, num_steps=200000, use_custom_num_steps=False*) [исходный код]

Параметры:
- **folder_dataset_name** – Название папки, в которой хранятся изображения и аннотации. По умолчанию - images_data.
- **model_number** – номер модели из файла All_Models, которая будет использоваться для обучения.
- **batch_size** – Размер бачей. По умолчанию 12.
- **num_steps** – Количество шагов для обучения. По умолчанию - максимум.
- **use_custom_num_steps** – Использовать ли параметр num_steps, введенный пользователем, или определить количестdo шагов внутри кода в зависимости от модели и размера датасета.

class_text_to_int(*row_label, d*) [исходный код]

Возвращает номер класса

Параметры:
- **row_label** (*str*) – название класса
- **d** (*dict*) – словарь из классов и их нумерации

Результат: номер класса

copy_pipeline() [исходный код]

Результат: считанный файл pipeline.config

create_annot_csv(*path_to_dataset*) [исходный код]

Создает и возвращает датафрейм, содержащий информацию о датасете из изображений. В случае, если csv с информауией уже существует, возвращает датафрейм с ним. Иначе - создает его из xml-файлов

Параметры: **path_to_dataset** (*str*) – путь к папке /images_data/all_images_data

Результат: датафрейм с информацией обо всех объектах датасета

create_pipeline_config(*s, annotations*) [исходный код]

Создание файла pipeline.config и замена необходимых строк на необходимые для конкретной задачи параметры.

Параметры: **s** (*str*) – содержимое файла pipeline.config
annotations (*list*) – список классов для детектирования
Результат:

create_tf_example(*group*, *path*, *annotations*) [исходный код]

Функция для создания tf_example (формат данных для тензорфлоу) из датасета

Параметры: **group** – строки из датафрейма для создания формата для object detection
path (*str*) – путь к папке /images_data
annotations (*list*) – список классов для детектирования
Результат: tf_example, формат хранения данных для обучения и инференса

label_map(*annotations*) [исходный код]

Создает файл с описанием классов формата .pbtxt

Параметры: **annotations** (*list*) – список классов для детектирования

write_to_record(*annot*, *annotations*) [исходный код]

Преобразование аннотаций в формат TFRecord для обучения с помощью Tensorflow Object Detection

Параметры: **annot** – датафрейм с информацией обо всех объектах датасета
annotations (*list*) – список из классов для детектирования

xml_to_csv(*path*) [исходный код]

Создает датафрейм из имеющихся в папке /images_data файлов формата .xml

Параметры: **path** (*str*) – путь до папки с xml файлами
Результат: датафрейм из данных о размеченных изображениях

Класс auto_object_detection_colab

class **ObjectDetection**(*folder_dataset_name='images_data'*, *model_number=20*, *batch_size=12*, *num_steps=200000*, *use_custom_num_steps=False*) [исходный код]

__init__(*folder_dataset_name='images_data'*, *model_number=20*, *batch_size=12*, *num_steps=200000*, *use_custom_num_steps=False*) [исходный код]

Параметры: **folder_dataset_name** – Название папки, в которой хранятся изображения и аннотации. По умолчанию - images_data.
model_number – номер модели из файла All_Models, которая будет использоваться для обучения.
batch_size – Размер бачей. По умолчанию 12.
num_steps – Количество шагов для обучения. По умолчанию - максимум.
use_custom_num_steps – Использовать ли параметр num_steps, введенный пользователем, или определить количестdo шагов внутри кода в зависимости от модели и размера датасета.

class_text_to_int(*row_label*, *d*) [исходный код]

Возвращает номер класса

Параметры: **row_label** (*str*) – название класса
d (*dict*) – словарь из классов и их нумерации
Результат: номер класса

copy_pipeline() [исходный код]

Результат: считанный файл pipeline.config

create_annot_csv(*path_to_dataset*) [исходный код]

Создает и возвращает датафрейм, содержащий информацию о датасете из изображений. В случае, если csv с информауией уже существует, возвращает датафрейм с ним. Иначе - создает его из xml-файлов

Параметры: **path_to_dataset** (*str*) – путь к папке /images_data/all_images_data
Результат: датафрейм с информацией обо всех объектах датасета

create_pipeline_config(*s*, *annotations*) [исходный код]

Создание файла pipeline.config и замена необходимых строк на необходимые для конкретной задачи параметры.

Параметры: **s** (*str*) – содержимое файла pipeline.config
annotations (*list*) – список классов для детектирования
Результат:

create_tf_example(*group*, *path*, *annotations*) [исходный код]

Функция для создания tf_example (формат данных для тензорфлоу) из датасета

Параметры: **group** – строки из датафрейма для создания формата для object detection
path (*str*) – путь к папке /images_data
annotations (*list*) – список классов для детектирования
Результат: tf_example, формат хранения данных для обучения и инференса

label_map(*annotations*) [исходный код]

Создает файл с описанием классов формата .pbtxt

Параметры: **annotations** (*list*) – список классов для детектирования

write_to_record(*annot*, *annotations*) [исходный код]

Преобразование аннотаций в формат TFRecord для обучения с помощью Tensorflow Object Detection

Параметры: **annot** – датафрейм с информацией обо всех объектах датасета
annotations (*list*) – список из классов для детектирования

xml_to_csv(*path*) [исходный код]

Создает датафрейм из имеющихся в папке /images_data файлов формата .xml

Параметры: **path** (*str*) – путь до папки с xml файлами
Результат: датафрейм из данных о размеченных изображениях

Класс inference

class **Inference**(*images_list*, *saved_model_path*, *label_map_path*) [исходный код]

__init__(*images_list*, *saved_model_path*, *label_map_path*) [исходный код]

detection_result() [исходный код]

Детектирует объекты на изображениях, записывает результаты в папку result, информауию о координатах записывает в result/output.csv :return:

get_coordinates(*threshold*, *output_dict*, *image_width*, *image_height*) [исходный код]

Параметры: **threshold** – уровень доверия
output_dict – предсказанные параметры
image_width –
image_height –
Результат: словарь из класса и его относительных координат

load_image_into_numpy_array(*image_path*) [исходный код]

Параметры: **path** (*str*) –
Результат: Массив изображения

Класс ScenarioDetector

class **ScenarioDetector**(*model_name='centernet_hg104_1024x1024_coco17_tpu-32'*, *model_date='20200711'*, *sequence_of_actions=None*, *images_list=None*, *label_filename='mscoco_label_map.pbtxt'*) [исходный код]

__init__(*model_name='centernet_hg104_1024x1024_coco17_tpu-32'*, *model_date='20200711'*, *sequence_of_actions=None*, *images_list=None*, *label_filename='mscoco_label_map.pbtxt'*) [исходный код]

Параметры: **model_name** – название модели
model_date – дата создания модели
sequence_of_actions – последовательность действий
images_list – список из изображений
label_filename – путь к файлу с метками

check_sequence(*class_number*, *predict_class*, *res*, *diff_classes*) [исходный код]

Проверка корректности последовательности задетектированных объектов

Параметры: **class_number** (*int*) – ожидаемый индекс списка последовательности объектов
predict_class (*list*) – задетектированные на кадре объекты
res (*list*) – список из True/False полученного объекта последовательности
diff_classes (*set*) – все предсказанные классы

Результат:

detect(*image*, *model*, *count*) [исходный код]

Детекция изображения

Параметры: **image** – путь к изображению
model – путь к модели
count – номер изображения
label_offset –

Результат:

download_labels() [исходный код]

Загружает файл с объектами и метками

Результат: путь до файла с метками

download_model() [исходный код]

Загружает модель

Результат: путь до загруженной модели

get_classes_from_frame(*threshold*, *output_dict*) [исходный код]

Возвращает список предсказанных классов

Параметры: **threshold** – уровень доверия
output_dict – предсказанные параметры

Результат:

load_image_into_numpy_array(*image_path*) [исходный код]

Параметры: **path** (*str*) –
Результат: Массив изображения

run_inference_for_single_image(*model*, *image*) [исходный код]

Инференс для одного кадра

Параметры: **model** – загруженная модель
image – изображение

Результат: output_dict

run_video(*path_to_video*) [исходный код]

Трекинг видео

Параметры: **path_to_video** –
Результат:

Класс IoU

class **IoU**(*path='result/outputs.csv'*) [исходный код]

__init__(*path='result/outputs.csv'*) [исходный код]

bb_intersection_over_union(*detected*, *real*) [исходный код]

Считает метрику IoU между двумя детекциями.

Параметры: **detected** – Датафрейм из предсказанных координат определенного объекта
real – Датафрейм из реальных координат определенного объекта

Результат: метрика IoU

xml_get_bnd_boxes(*path*) [исходный код]

Создает из xml файлов датафрейм из координат

Параметры: **path** – Путь к xml файлу
Результат: Датафрейм из координат

Модуль fullyfunctionality

launch(*model*, *labelmap*, *video_path*) [исходный код]

Запуск видео с задетектированными объектами

load_model(*model_path*) [исходный код]

Параметры: **model_path** (*str*) – путь к модели
Результат: загруженная модель

run_inference(*model*, *category_index*, *cap*) [исходный код]

Параметры: **model** – модель
category_index – индекс
cap – cv2.VideoCapture

run_inference_for_single_image(*model*, *image*) [исходный код]

Инференс для одного кадра

Параметры: **model** – загруженная модель
image – изображение
Результат: output_dict

Модуль video_stream

run_inference(*model*, *cap*) ¶ [исходный код]

Запускает видеокамеру с обнаружением объектов.

Параметры: **model** – Путь к модели
cap – cv2.VideoCapture(0)

run_inference_for_single_image(*model*, *image*) [исходный код]

Параметры: **model** – Путь к модели
image – Изображение
Результат: Словарь с наименованием класса и его положением на изображении

Модуль resize_images

preprocess_resize(*target_width*) [исходный код]

Меняет размер изображений

Параметры: **target_width** – Размер в пикселях
Результат: Изображение

Indices and tables

Алфавитный указатель
Состав модуля
Поиск