
Dataset Stats
Alain

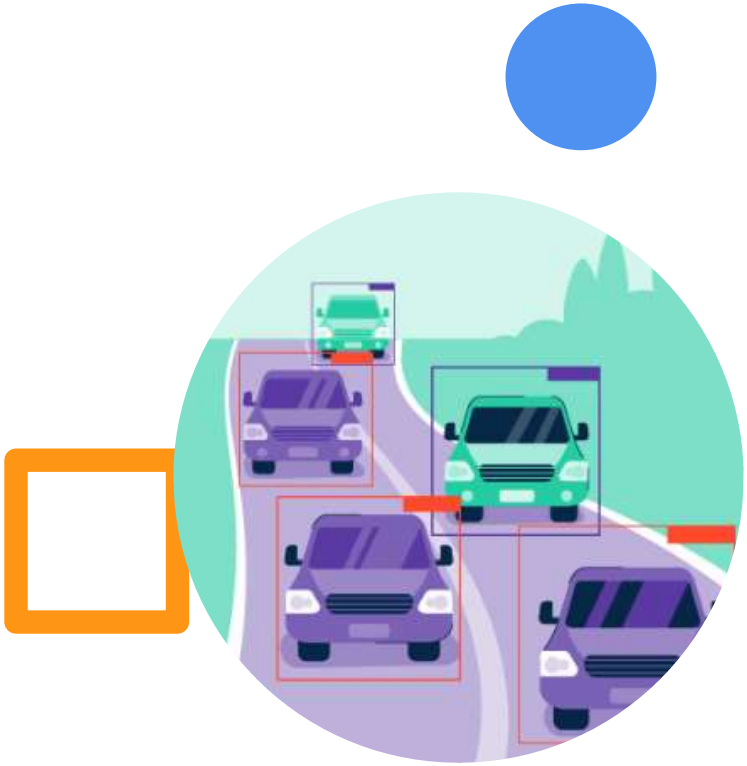
Agenda

What we are doing
Code Access + Structure
Getting started
Organization
Contribute to code
Conclusion

What we are doing

Vision Tasks Based Dataset Stats

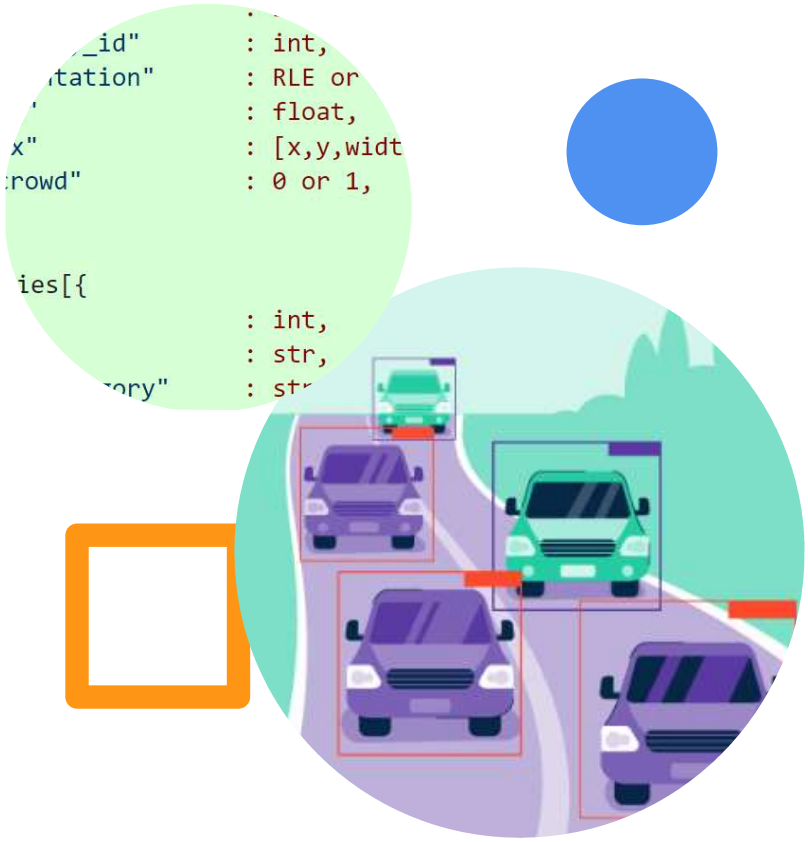
- Object Detection
- Instance Segmentation
- Multiple Object Tracking (MOT)
- Video Instance Segmentation



What we are doing

We are challenged with:

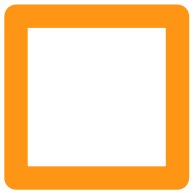
- Variety
- Standardized Structure
- Comprehensive per task datasets comparison



Task	Detection Based	Instance Segmentation	Multi Object Tracking	Video Instance Segmentation
Dataset	✓ COCO ✓ SkyData ✓ VisDrone ✓ KAIST ✓ VHR-10 ✓ DOTA ✓ VEDAI ✓ KITTI	✓ COCO ✓ SkyData ✓ VHR-10	✓ SkyData ✓ VisDrone-MOT ✓ MOT-17 ✓ MOT-20 ✓ DanceTrack ✓ TAO	✓ SkyData ✓ Youtube-VIS 2019 ✓ Youtube-VIS 2021

Access the Code

```
# Get REPO
#1. clone and setup up the repo
!git clone https://github.com/ozerlabs-proxy/vision-datasets-
stats.git

#2. cd into the repo
cd vision-datasets-stats


#3.
#we require conda to be installed
#alternatively you can any other env
conda env create -f environment.yaml
conda activate VisionStats

#4. follow along the notebooks
```



Codebase Structure

```
datasets-stats
├── bases
│   ├── __init__.py
│   ├── coco.py
│   ├── skydata.py
│   ... # each responsible to load and parse a dataset
├── data
│   ├── coco
│   │   ├── annotations
│   │   ├── ...
│   ├── skydata
│   │   ├── annotations
│   │   ├── ...
│   ... # data for datasets (images, annotations, etc)
├── notebooks
├── scripts
│   │   ├── convert_dota_to_coco.py
│   │   ├── convert_visdrone_to_coco.py
│   │   ... # scripts to convert from one format to another
├── utils
│   ├── __init__.py
│   ├── conversion_dota_tools.py
│   ├── conversion_visdrone_tools.py
│   ├── tracking_stats_tools.py
│   ... # contains visualization, conversion, stats,tools
├── generate_stats.py
├── README.md
├── requirements.txt
└── ...
```

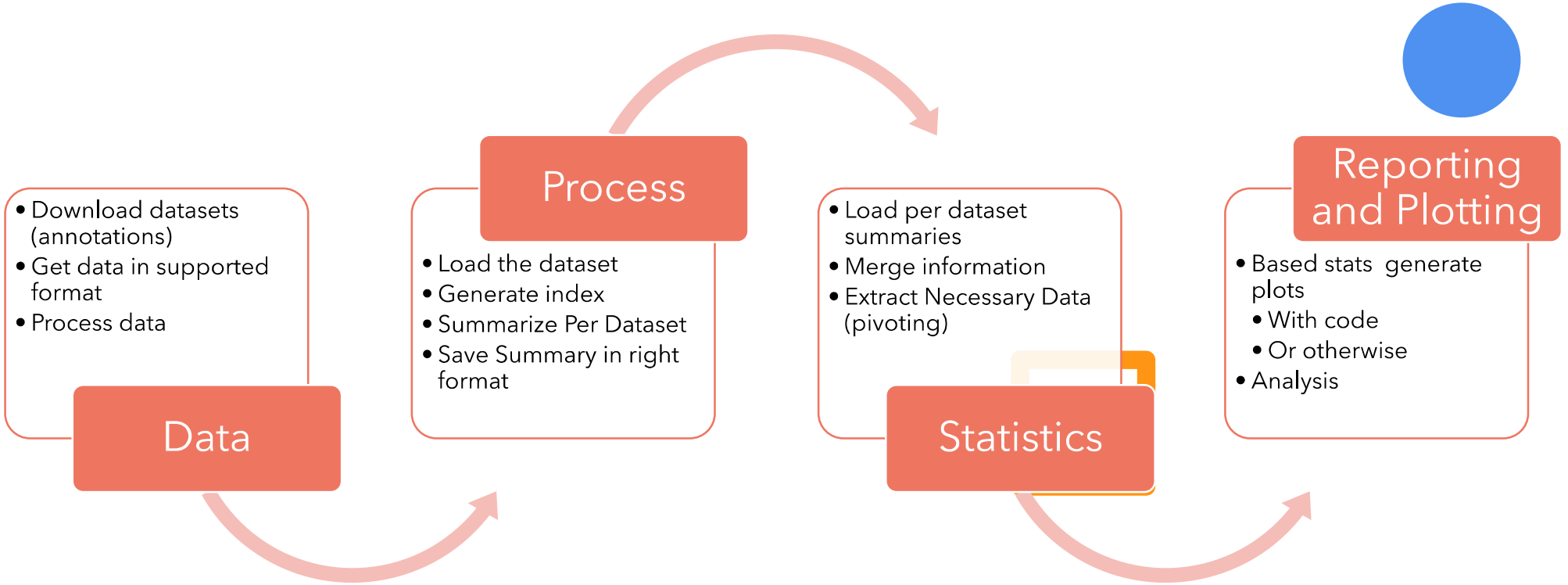
Getting Started

```
"""
We will need sample data to follow along, download coco2014 and put it in the right
place
format is as follows:
"""

#!/bin/bash

mkdir -p "data/coco_2014"
cd "data/coco_2014"

# download coco 2014 annotations
wget -c  http://images.cocodataset.org/annotations/annotations_trainval2014.zip
wget -c  http://images.cocodataset.org/annotations/image_info_test2014.zip

unzip annotations_trainval2014.zip
unzip image_info_test2014.zip

rm annotations_trainval2014.zip
rm image_info_test2014.zip
```

Organization

Steps -- Purpose

Steps


Data
• Download datasets (annotations)
• Get data in supported format
• Process data
Process
• Load the dataset
• Generate index
• Summarize Per Dataset
• Save Summary in right format
Statistics
• Load per dataset summaries
• Merge information
• Extract Necessary Data (pivoting)
Reporting and Plotting
• Based stats generate plots
• With code
• Or otherwise
• Analysis

Steps

- Download datasets (annotations)
- Get data in supported format
- Process data

Data

Aims at getting data. If not formatted use existing converters or (create new one) to get the data in right format.

```python
# Path to the images and annotations
kaist_path = Path('data/kaist_pedestrian/')
annotations_dir = kaist_path / 'annotations-vbb'
_outputdir = kaist_path / "converted_annotations"

if not annotations_dir.exists():
    print(f"\n[INFO] vbb annotations not found...")
    print(f"[INFO] Downloading annotations...")

    try:
        os.makedirs(annotations_dir)
        os.system( 'curl -O http://multispectral.kaist.ac.kr/pedestrian/data-kaist
        os.system( 'unzip -d %s/annotations-vbb -q annotations-vbb.zip' % kaist_pa
    except Exception as e:
        print(f"\n[ERROR] {e}")
        print(f"[ERROR] Downloading annotations failed.")
        exit(1)

# convert kaist to coco format
print(f"\n[INFO] Converting kaist to coco format...")
parsed_annotations = conversion_kaist_tools.convert_kaist_to_coco(annotations_dir)
print(f"\n[INFO] Converting kaist to coco format completed.")

# # save the json file
utilities.save_json(data=parsed_annotations,
                    save_dir=str(_outputdir),
                    file_name="kaist_converted_to_coco_format.json")
```

Steps

- Load the dataset
- Generate index
- Summarize Per Dataset
- Save Stats in right format

Process

Once Data is formatted.

We load, create Index, process and extract informative attributes i.e categories, track lengths ...

```python
## 3. YoutubeVis2021 database

TASK='VIS'

from bases.youtube_vis_dataset import YoutubeVisDataset
from pathlib import Path

dataset_year = 2021
dataset_stem = f"ytvis_{dataset_year}"
split = "train"
subset = f"instances_{split}_sub"
annotiation_file = Path(f"./data/{dataset_stem}/annotations/{subset}.json")
D = YoutubeVisDataset(annotation_file=str(annotiation_file))

# generate and load stats
D.generate_dataset_statistics()

# save the stats
D.save_dataset_statistics(save_path = f"./summaries/{TASK}",
                          dataset_name = f"{dataset_stem}",
                          file_name = f"{subset}_stats.json"
                          )
print(f"[INFO] Saved")
```

Steps

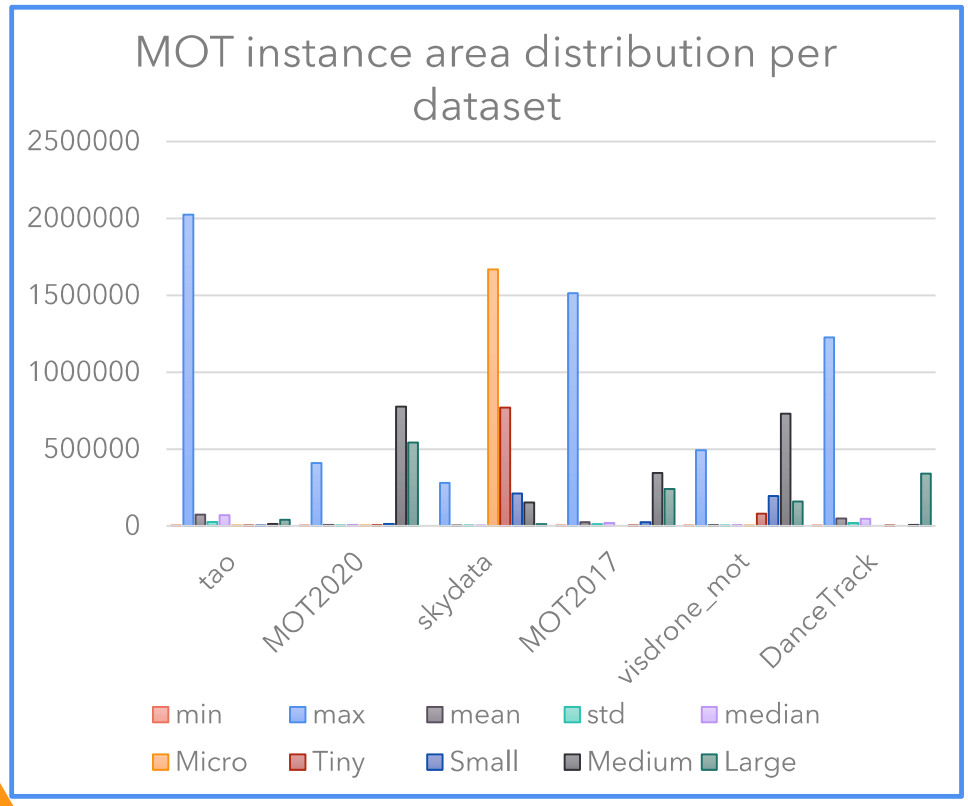
- Load per dataset summaries
- Merge information
- Extract Necessary Data (pivoting)

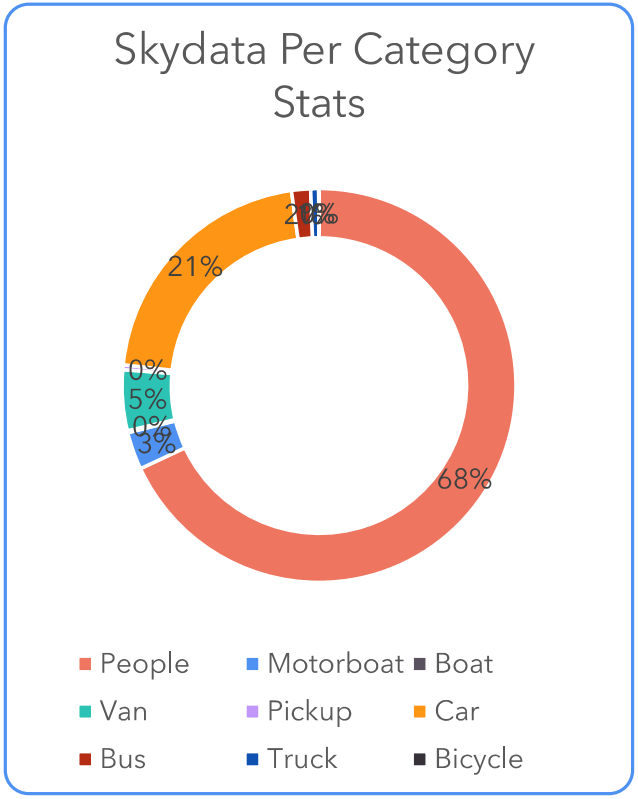
Statistics

Load all previously saved stats per datasets per task. Combine them and generate overall summaries and plots.

```
# plot generals and save stats

## global_summary_plain_value_cols_df save to csv
stats_tools.save_df_to_csv(df=global_summary_plain_value_cols_df,
                           save_path=summaries_path/'all_datasets',
                           file_name='_plain_value_global.csv')


## global_summary_images_stats save to csv
stats_tools.summarize_global_images_plot_and_save(global_summary_images_stats=global_summar
                                                  save_path=summaries_path/'all_datasets',
                                                  file_name='_images_stats_global')

## global_summary_masks_stats save to csv
stats_tools.summarize_global_summary_masks_stats_plot_and_save(global_summary_masks_stats=g
                                                  save_path=summaries_path/'all_datasets',
                                                  file_name='_masks_stats_global')

# ## global_areas_ranges_stats save to csv
stats_tools.summarize_global_areas_ranges_stats_plot_and_save(global_areas_ranges_stats_df=
                                                  save_path=summaries_path/'all_datasets',
                                                  file_name='_areas_ranges_stats_global')

# ## global_areas_ratios_general_stats save to csv
stats_tools.summarize_global_areas_ratios_general_stats_plot_and_save(stats=global_areas_ra
                                                                      save_path=summaries_pat
                                                                      file_name='_areas_ratio
```

Reporting


MOT instance area distribution per dataset
2500000
2000000
1500000
1000000
500000
0
tao
MOT2020
skydata
MOT2017
visdrone_mot
DanceTrack
min
max
mean
std
median
Micro
Tiny
Small
Medium
Large


Skydata Per Category Stats
21%
0%
5%
0%
3%
68%
People
Motorboat
Boat
Van
Pickup
Car
Bus
Truck
Bicycle

Note

The whole process generates stats for each dataset individually and gives informative summaries which is task dependent.

Detection

- Dataset Size
- Categories
- Area

MOT

- Video Count
- Categories
- Instance Count

Vis

- Track length
- Seg Area
- Tracks per cat

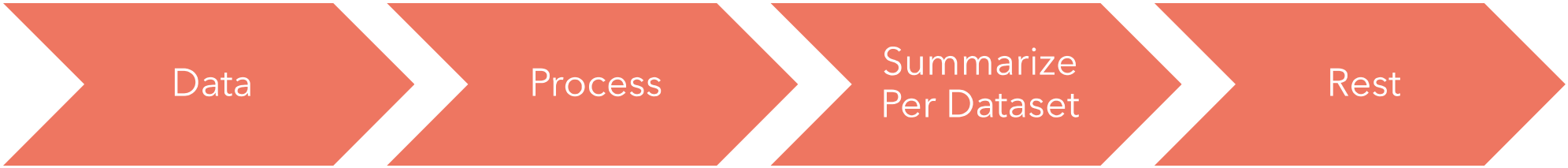
Note

Although we get informative plots and summaries from the whole process, It is recommended to consider taking the summary csv files generated in the process and use advanced tools for better and effortless visualizations.

This is can be easily done by simply importing those files in excel.


Contribute
Add feature - extend the code

Support Dataset for existing Tasks



- Download datasets and get it in the right directory.
- You will need to get the data into coco format for most tasks.
- For this you can build on existing converters in tools folder.

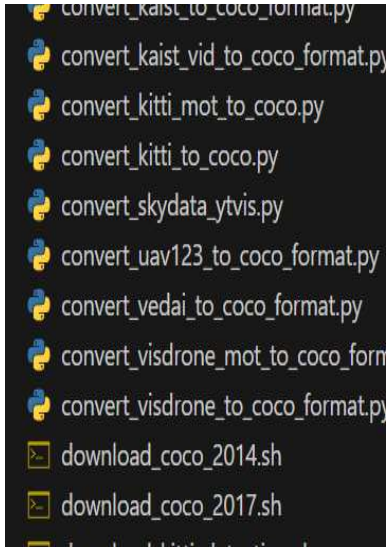
- You will need build on, use, or extend existing dataset loaders and parsers in bases folder.

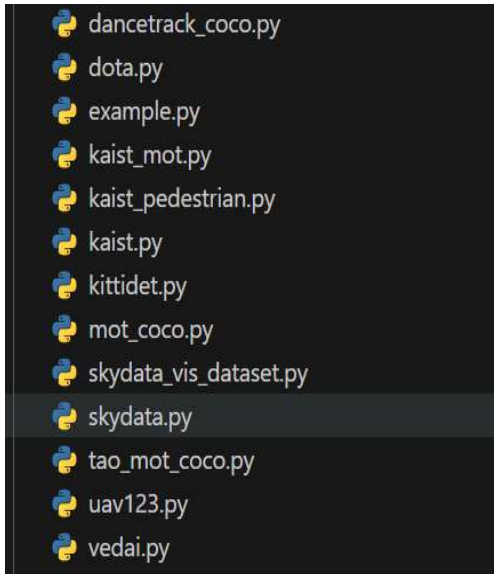
- Load the dataset, generate index, making sure that you expose required fields in the stats json file generated inside the loader in previous step.

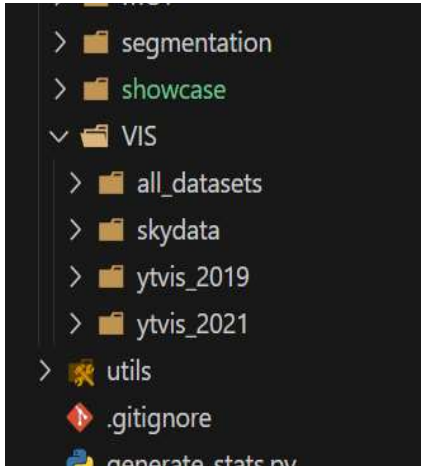
- If all steps done right. Head to the corresponding script for generating statistics and run it. It will automatically include new dataset.

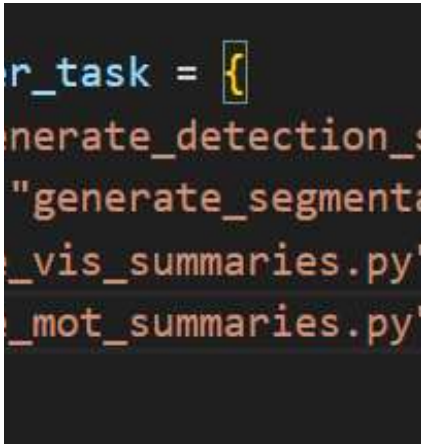
Support Dataset for existing Tasks

Data → Process → Summarize Per Dataset → Rest



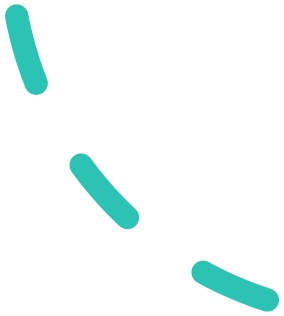






Summary

In addition to the stats generated this codebase provides building blocks for dataloaders of different datasets and different computer vision tasks.



Supplements

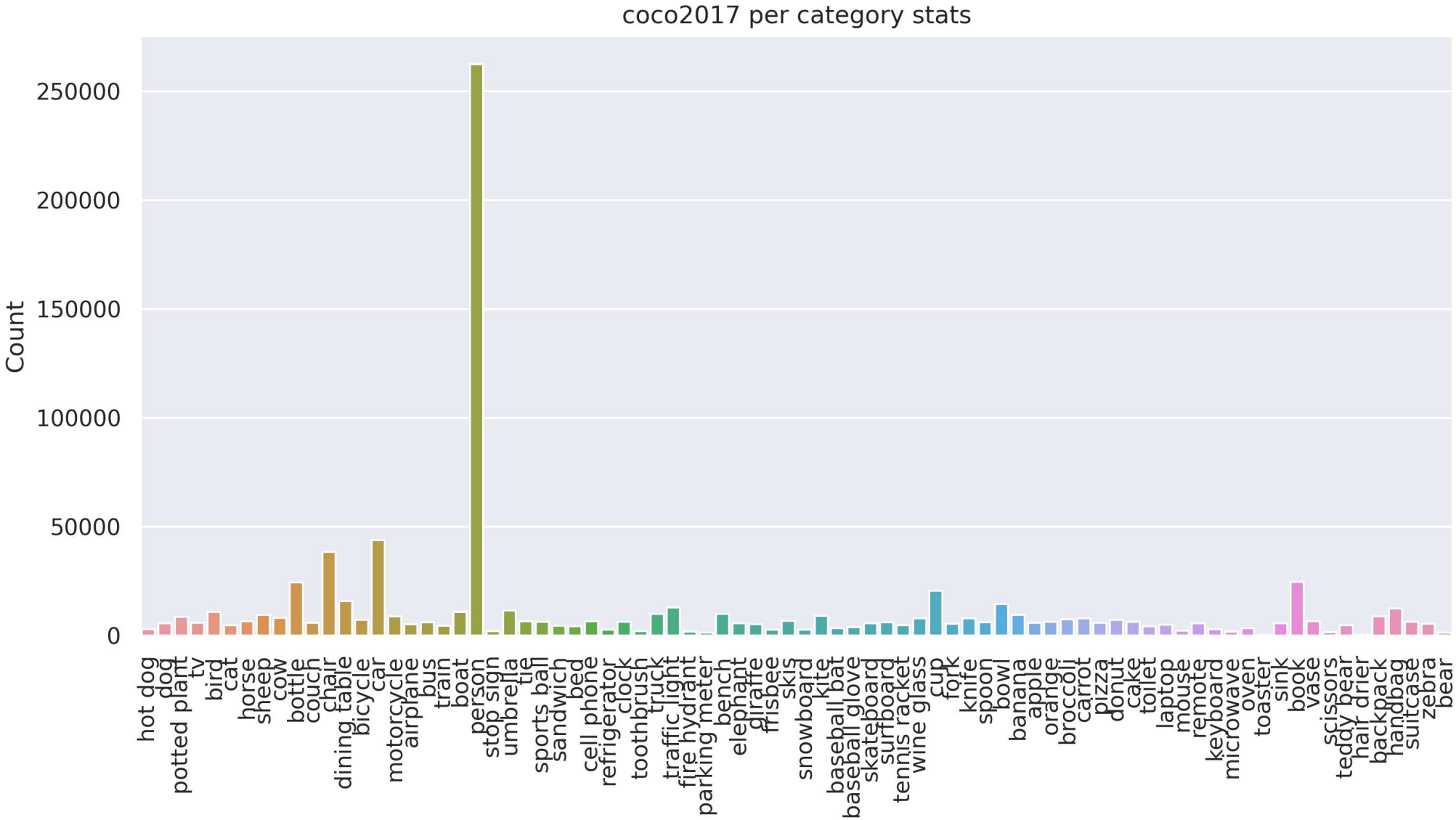
Plots and Visuals

Object Detection Task General Stats

dataset_name	nnotations_coun	created_by	images_count	task	dataset_size	cat_count
DOTA	268627	DOTA	1830	detection	1830	18
VisDrone	353550	VisDrone	6471	detection	6471	12
KAIST Pedestrian	23309	KAIST	17498	detection	95324	3
KITTI	51865	KITTI	7481	detection	7481	9
SkyData	2818759	SkyData	8085	detection	8085	9
VHR10	3921	VHR10	650	detection	650	10
Vedai	3757	Vedai	1246	detection	1246	15
COCO	860001	Microsoft	118287	detection	118287	80
COCO	604907	Microsoft	82783	detection	82783	80
KAIST_roboflow	16699	KAIST	3296	detection	3296	4
UAV123	112578	UAV123	112578	tracking	112578	1

Object Detection Task Sizes Stats

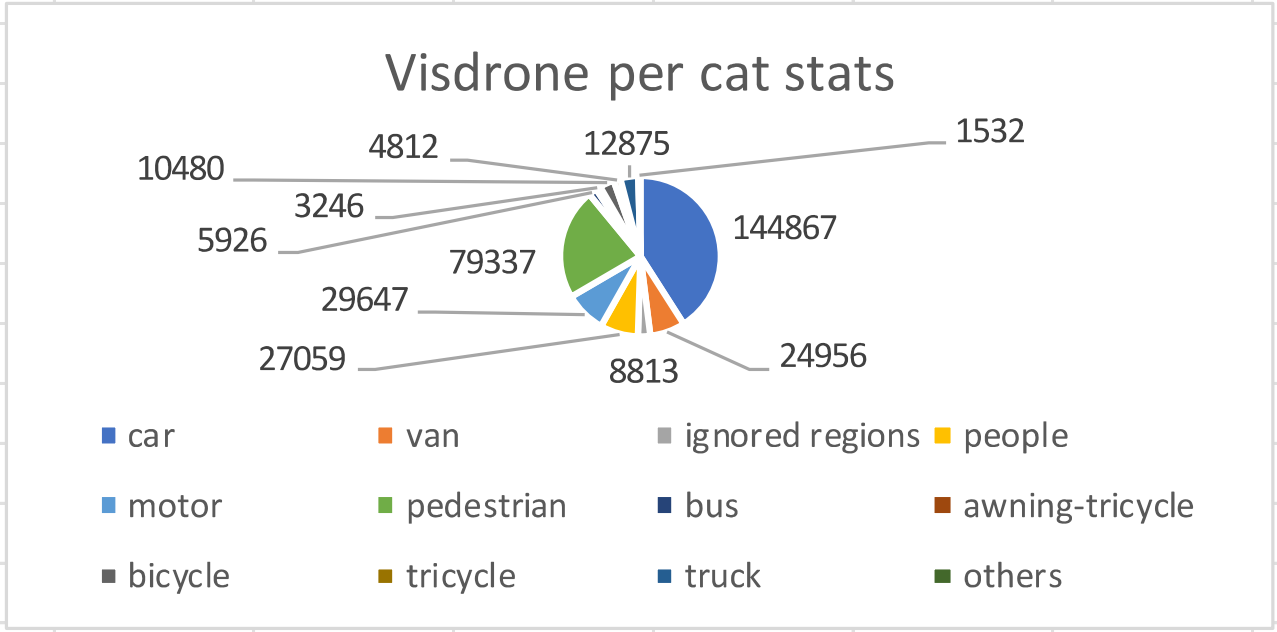
dataset_name	Micro	Tiny	Small	Medium	Large
DOTA	94209	76845	44915	44857	7801
visdrone	52012	92736	67882	120961	19959
kaist_pedestrian	0	1	12404	87454	8273
kitti	912	5255	8053	25524	12121
skydata	1669426	770670	212684	154069	11910
vhr_10	0	105	641	2894	281
vedai	0	72	147	880	2658
coco2017	105297	143308	107735	295163	208498
coco2014	74491	101021	75468	207797	146130
kaist_roboflow	44	1261	2344	9987	3063
uav123	5588	7909	11622	65275	22184


coco2017 per category stats
Count
250000
200000
150000
100000
50000
0
hot dog
dog
potted plant
tv
bird
cat
horse
sheep
cow
bottle
couch
chair
dining table
bicycle
car
motorcycle
airplane
bus
train
boat
person
stop sign
umbrella
tie
sports ball
sandwich
bed
cell phone
refrigerator
clock
toothbrush
truck
traffic light
fire hydrant
parking meter
bench
elephant
giraffe
frisbee
skis
snowboard
kite
baseball bat
baseball glove
skateboard
surfboard
tennis racket
wine glass
cup
fork
knife
spoon
bowl
banana
apple
orange
broccoli
carrot
pizza
donut
cake
toilet
laptop
mouse
remote
keyboard
microwave
oven
toaster
sink
book
vase
scissors
teddy bear
hair drier
backpack
handbag
suitcase
zebra
bear

Segmentation Task General Stats

dataset_name	ann_count	im_count	task	dataset_size	cat_count
DOTA	268627	1830	seg	1830	18
SkyData	2818759	8085	seg	8085	9
VHR10	3921	650	seg	650	10
Vedai	3757	1246	seg	1246	15
COCO	860001	118287	seg	118287	80
COCO	604907	82783	seg	82783	80

Segmentation Task Sizes Stats



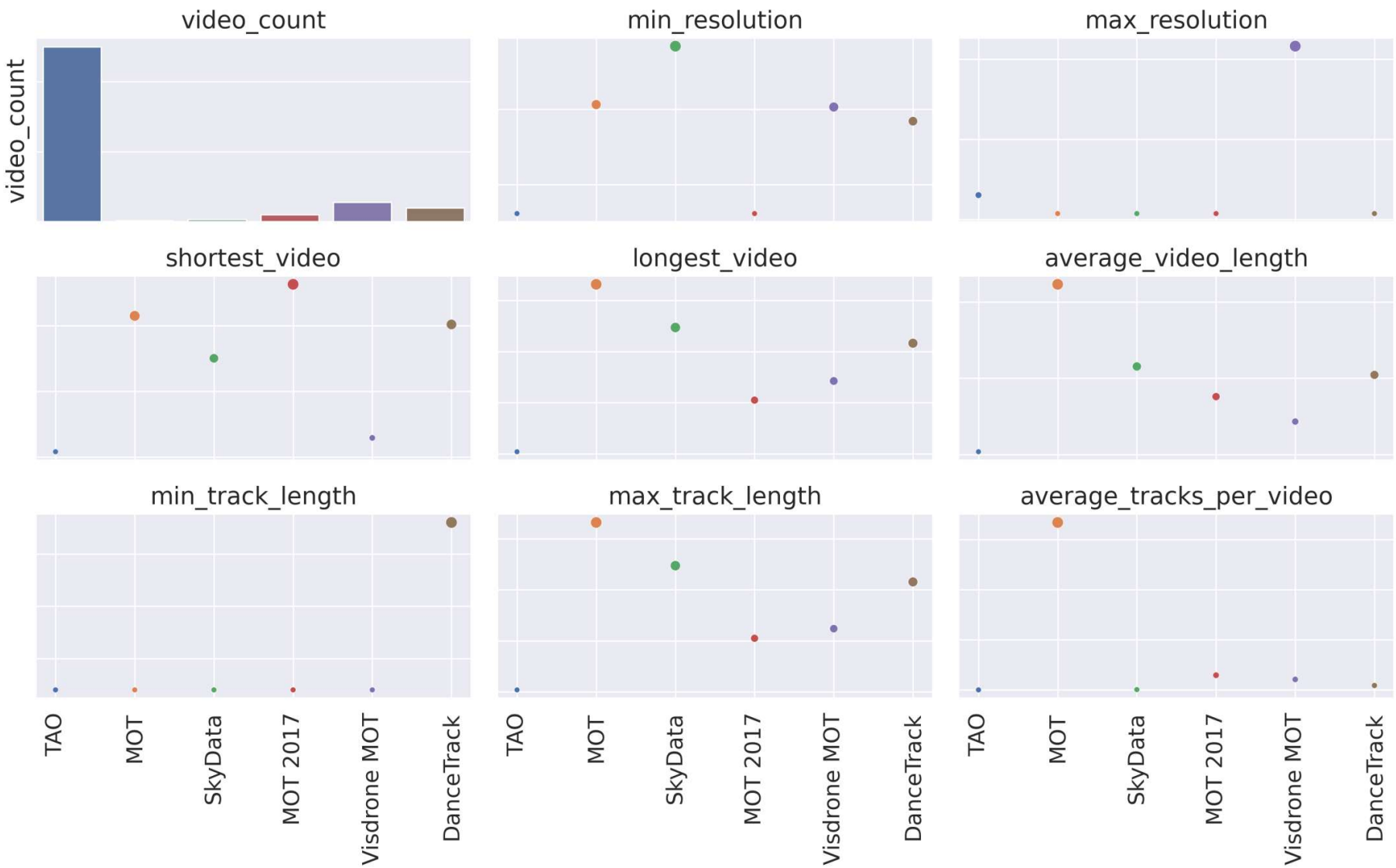
MOT Task General Stats

dataset_name	task	video_c	year	min_h	max_hei	min_wi	max_wid	shortest	longest_	average_	total_tra	average_	min_	max_tra	average_t
TAO	MOT	500	2020	480	1920	480	1920	16	41	36.548	2833	20.897	1	41	5.666
MOT	MOT	4		880	1080	1173	1920	429	3315	2232.8	2332	573.29	1	3315	334230
SkyData	Vis	7	2022	740	1080	1920	1920	300	2470	1155	3943	714.88	1	2470	563.286
MOT 2017	MOT	21		480	1080	640	1920	525	1050	759.43	2388	257.16	1	1050	29243
Visdrone MOT	MOT	56	2019	756	1530	1344	2720	58	1424	432.16	7089	158.82	1	1238	20881.4
DanceTrack	MOT	40		720	1080	1280	1920	403	2163	1044.9	419	832.77	9	2152	8723.25

Vis Task General Stats

dataset_name	total_tracks	average_track_l	min_trac	max_trac	average_t
skydata	3943	714.8767436	1	2470	563.2857
ytvis_2019	3774	25.73131955	1	36	1.686327
ytvis_2021	6283	27.9140538	1	72	2.104858

dataset_nam	min	max	mean	std	median	Micro	Tiny	Small	Medium	Large
skydata	0	279474	402.7806	75.00033	398.4344	1669426	770670	212684	154069	11910
ytvis_2019	101	921600	110127.5	19929.23	109407.5	28	480	1184	11822	83596
ytvis_2021	1	921600	72767.15	14240.16	72020.23	1331	3313	6430	44274	120036


Videos Stats per Dataset
video_count
min_resolution
max_resolution
shortest_video
longest_video
average_video_length
min_track_length
max_track_length
average_tracks_per_video
video_count
TAO
MOT
SkyData
MOT 2017
Visdrone MOT
DanceTrack