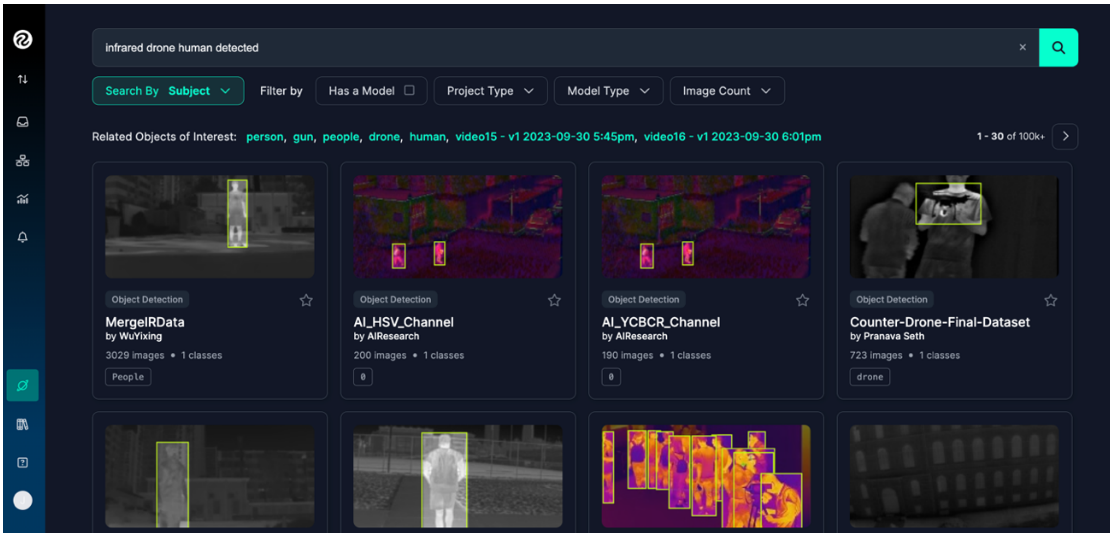
Step-by-Step Guide for Training a Model on Azure

1. Set Up Roboflow Account

- **Create a Roboflow Account**:
  - Visit Roboflow and sign up for an account if you don't already have one.
  - Verify your email and log in to your account.

2. Collecting and Transferring Data on Roboflow

- **Sign in to Roboflow**:
  - Search for the open dataset source that you need on the Roboflow Universe.
  - Example: Search for Infrared Human Detection Dataset.
  - Enter the Roboflow Universe page to find datasets related to human detection using infrared drone images, focusing on those with bounding box annotations.
  - Ensure the dataset includes images captured from different heights and angles to increase diversity.
  - Roboflow Universe



infrared drone human detected

Related Objects of Interest: person, gun, people, drone, human, video15 - v1 2023-09-30 5:45pm, video16 - v1 2023-09-30 6:01pm

1 - 30 of 100k+

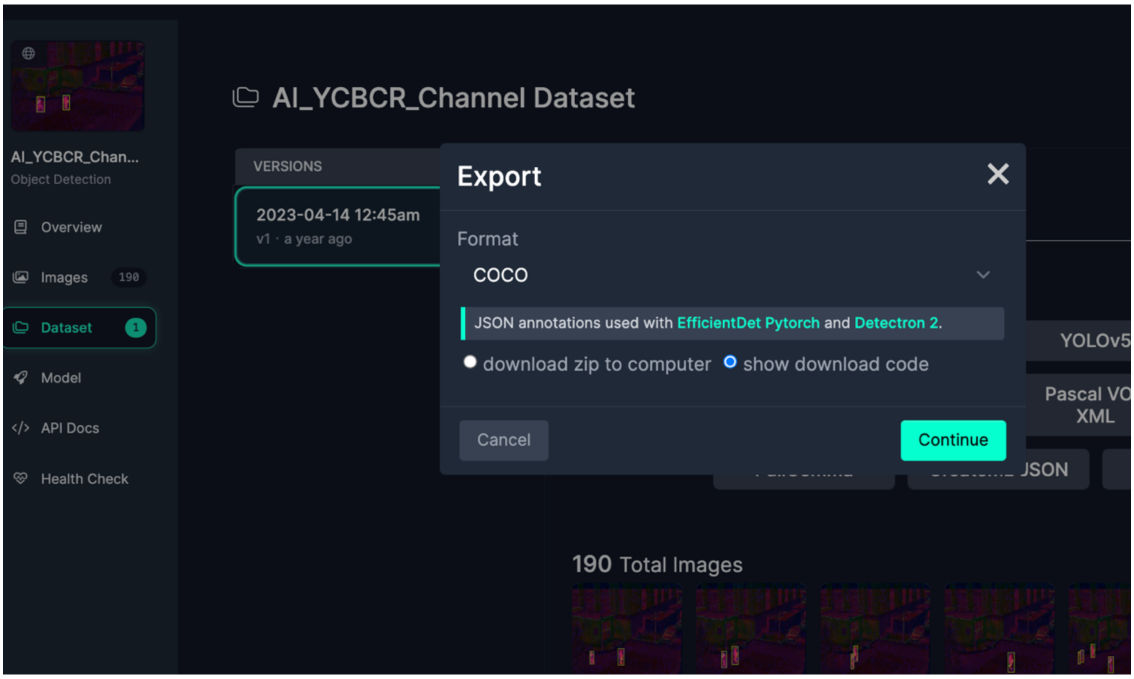
Object Detection
MergeIRData
by WuYixing
3029 images • 1 classes
People

Object Detection
AI_HSV_Channel
by AIResearch
200 images • 1 classes
0

Object Detection
AI_YCBCR_Channel
by AIResearch
190 images • 1 classes
0

Object Detection
Counter-Drone-Final-Dataset
by Pranava Seth
723 images • 1 classes
drone

- **Select and Export the Dataset:**
  - Choose a dataset with at least 5000 images. If a single dataset does not meet this requirement, consider combining multiple datasets.
  - Export the dataset in the COCO JSON format, which includes images and their corresponding bounding box annotations in a JSON file.
  - Download the dataset zip file to your local machine.



AI_YCBCR_Channel Dataset

AI_YCBCR_Chan...
Object Detection

VERSIONS
2023-04-14 12:45am
v1 · a year ago

Export

Format: COCO

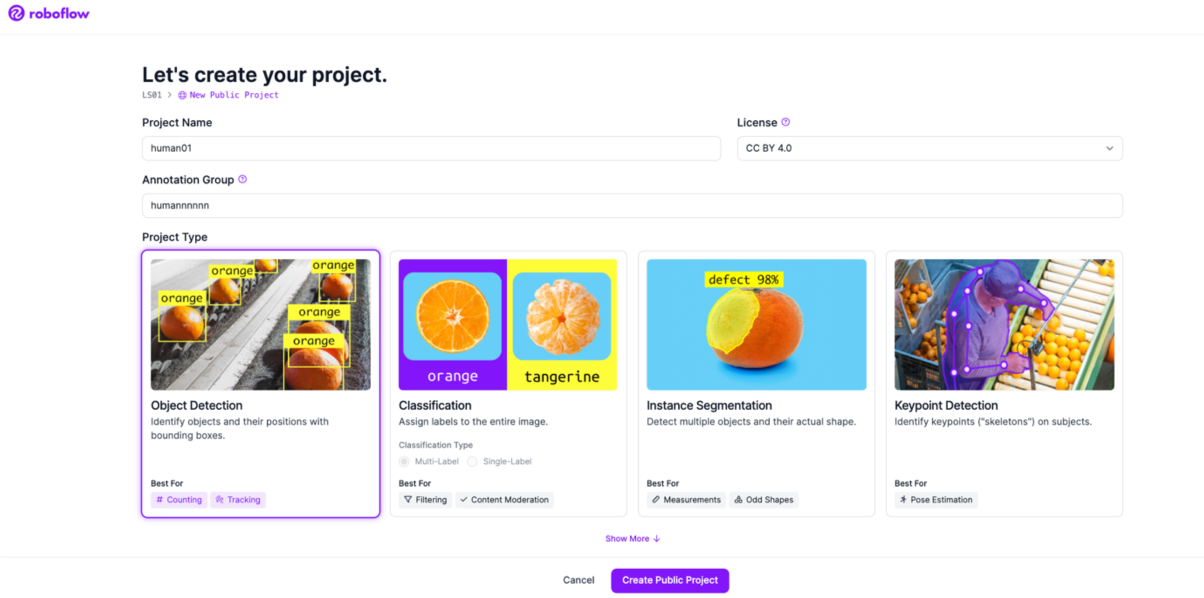
JSON annotations used with EfficientDet Pytorch and Detectron 2.

○ download zip to computer ● show download code

190 Total Images

- **Prepare for Data Transfer:**
  - Extract the downloaded zip file to ensure all images and annotations are correctly structured.
  - Verify that the extracted data includes image files and a single JSON file containing all the annotations.

- **Create a Project on Roboflow:**
  - Create a project by typing your project name and annotation group name and selecting the project type.



Let's create your project.
LS01 > New Public Project

Project Name: human01

License: CC BY 4.0

Annotation Group: humannnnnn

Project Type:

Object Detection (selected) — Identify objects and their positions with bounding boxes. Best For: Counting, Tracking

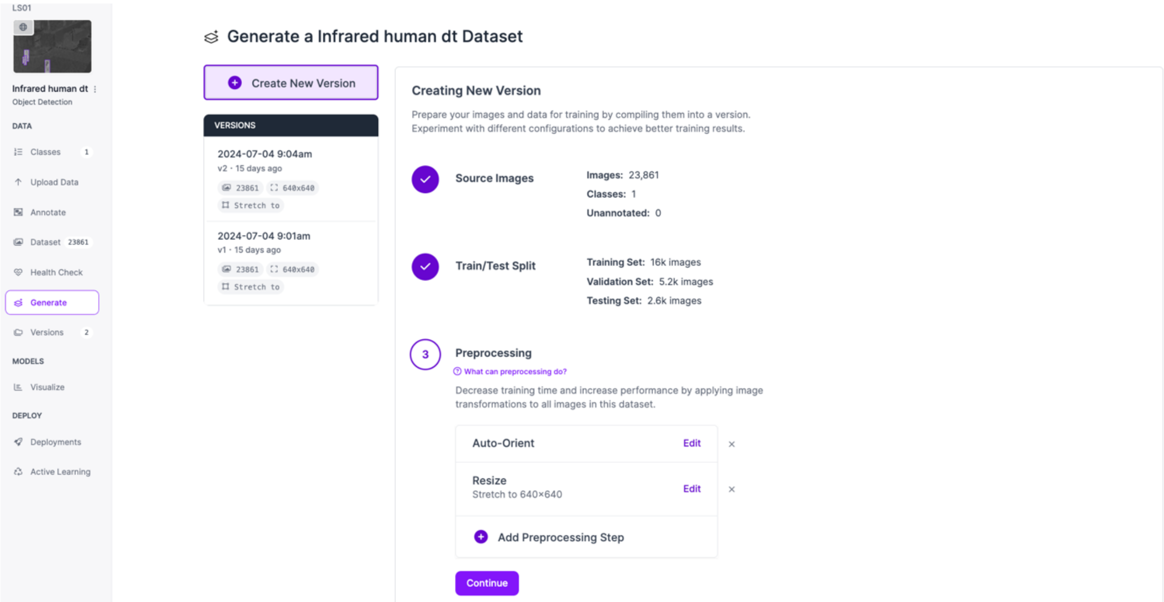
Classification — Assign labels to the entire image. Classification Type: ○ Multi-Label ○ Single-Label. Best For: Filtering, Content Moderation

Instance Segmentation — Detect multiple objects and their actual shape. Best For: Measurements, Odd Shapes

Keypoint Detection — Identify keypoints ("skeletons") on subjects. Best For: Pose Estimation

  -
  - Upload all the COCO dataset that you collected before.

- Make sure to change the classes if they are not correct and resize images to 640x640.



Generate a Infrared human dt Dataset

2024-07-04 9:04am
v2 · 15 days ago
23861 640x640
Stretch to

2024-07-04 9:01am
v1 · 15 days ago
23861 640x640
Stretch to

Creating New Version

Prepare your images and data for training by compiling them into a version. Experiment with different configurations to achieve better training results.

Source Images — **Images:** 23,861 **Classes:** 1 **Unannotated:** 0

Train/Test Split — **Training Set:** 16k images **Validation Set:** 5.2k images **Testing Set:** 2.6k images

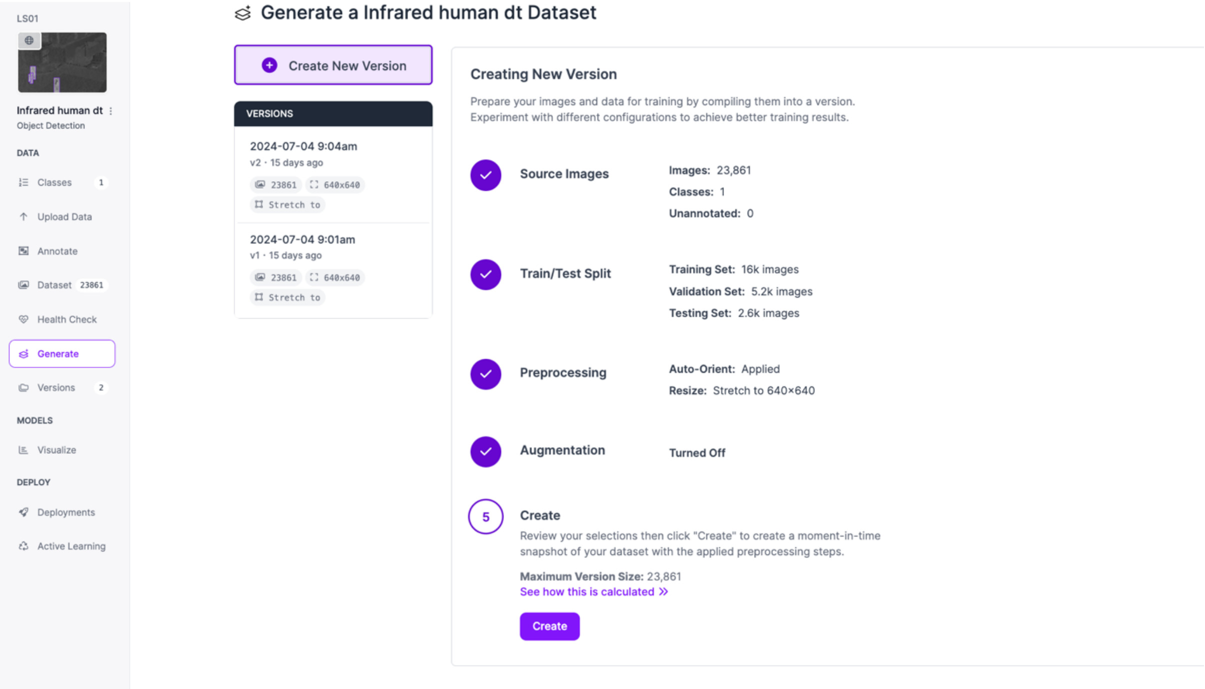
3 Preprocessing

What can preprocessing do?

Decrease training time and increase performance by applying image transformations to all images in this dataset.

Auto-Orient

Resize
Stretch to 640×640



Generate a Infrared human dt Dataset

2024-07-04 9:04am
v2 · 15 days ago
23861 640x640
Stretch to

2024-07-04 9:01am
v1 · 15 days ago
23861 640x640
Stretch to

Creating New Version

Prepare your images and data for training by compiling them into a version. Experiment with different configurations to achieve better training results.

Source Images — **Images:** 23,861 **Classes:** 1 **Unannotated:** 0

Train/Test Split — **Training Set:** 16k images **Validation Set:** 5.2k images **Testing Set:** 2.6k images

Preprocessing — **Auto-Orient:** Applied **Resize:** Stretch to 640×640

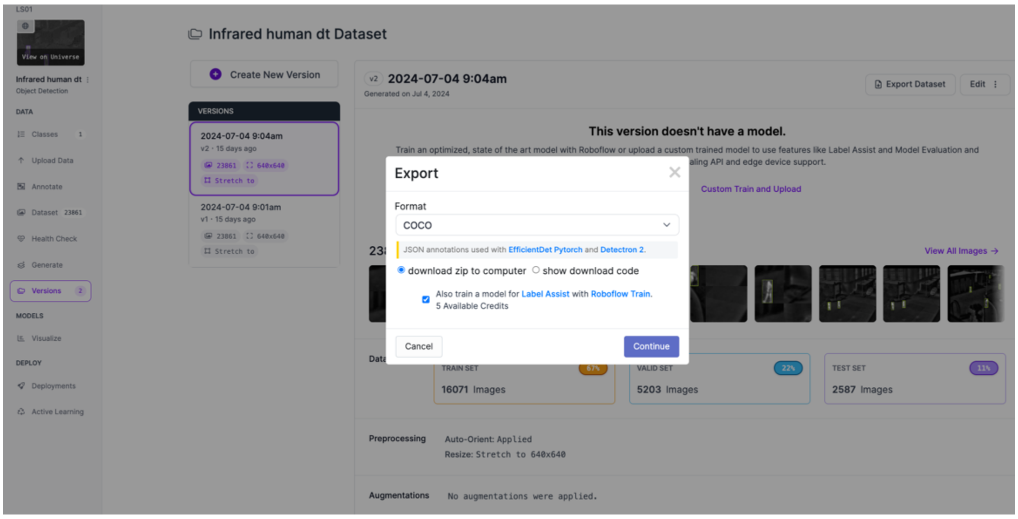
Augmentation — Turned Off

5 Create

Review your selections then click "Create" to create a moment-in-time snapshot of your dataset with the applied preprocessing steps.

Maximum Version Size: 23,861
See how this is calculated »

- Export the dataset to your local machine.

- 

  Infrared human dt Dataset

  2024-07-04 9:04am
  v2 · 15 days ago
  23861 640x640
  Stretch to

  2024-07-04 9:01am
  v1 · 15 days ago
  23861 640x640
  Stretch to

  v2 2024-07-04 9:04am
  Generated on Jul 4, 2024

  This version doesn't have a model.

  Train an optimized, state of the art model with Roboflow or upload a custom trained model to use features like Label Assist and Model Evaluation and …aling API and edge device support.

  Export

  Format: COCO

  JSON annotations used with EfficientDet Pytorch and Detectron 2.

  ● download zip to computer ○ show download code

  ☑ Also train a model for Label Assist with Roboflow Train.
  5 Available Credits

  TRAIN SET 67% 16071 Images
  VALID SET 22% 5203 Images
  TEST SET 11% 2587 Images

  Preprocessing: Auto-Orient: Applied
  Resize: Stretch to 640x640

  Augmentations: No augmentations were applied.

3. Login into Azure Server

- **Connect to Azure Server:**
  - Type the following command in the terminal:

    ```
    ssh -L 8080:localhost:8888 lifesparrow01@74.249.109.97
    ```

  - Enter the password: `Lifesparrow.5593`
  - Switch to the root user:

    ```
    sudo su - root
    ```

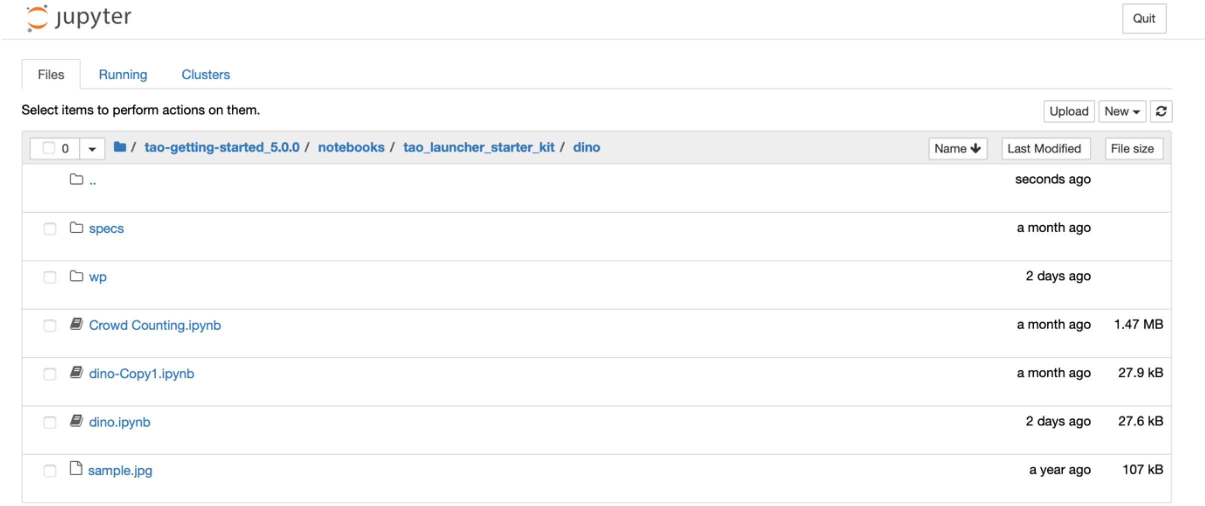
  - Set up the Python environment:

    ```
    export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
    source /usr/local/bin/virtualenvwrapper.sh
    workon launcher
    jupyter notebook --ip 0.0.0.0 --port 8888 --allow-root --
    NotebookApp.token=
    ```

  - Open your browser and access localhost:8080

4. Prepare the Project Directory

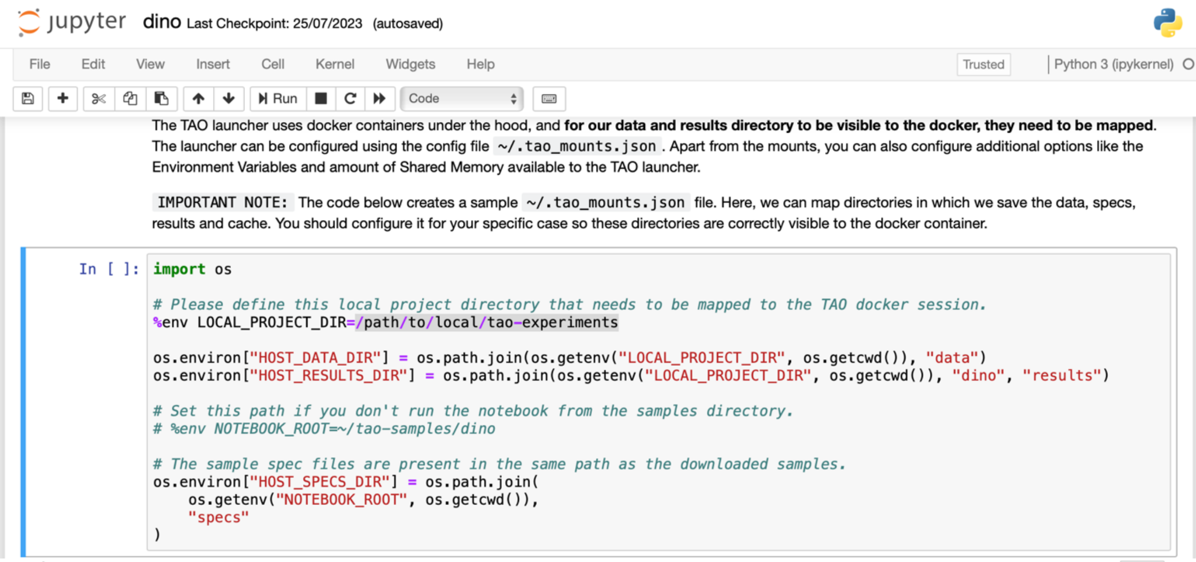
- **Set Up Project Directory**:
  - Open `dino.ipynb` under the `dino` folder.
  - Create a folder named `wp` under the `dino` folder.
  - Create a folder named `data` under the `wp` folder and a folder named `raw-data` under the `data` folder.



Select items to perform actions on them.

/ tao-getting-started_5.0.0 / notebooks / tao_launcher_starter_kit / dino

Name	Last Modified	File size
..	seconds ago	
specs	a month ago	
wp	2 days ago	
Crowd Counting.ipynb	a month ago	1.47 MB
dino-Copy1.ipynb	a month ago	27.9 kB
dino.ipynb	2 days ago	27.6 kB
sample.jpg	a year ago	107 kB

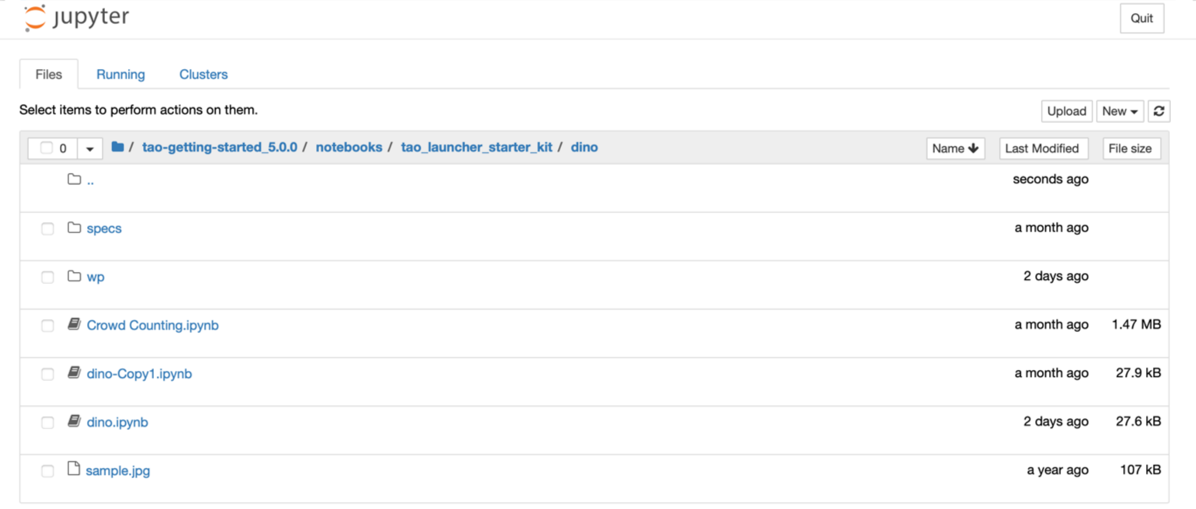

jupyter dino Last Checkpoint: 25/07/2023 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

The TAO launcher uses docker containers under the hood, and **for our data and results directory to be visible to the docker, they need to be mapped**. The launcher can be configured using the config file `~/.tao_mounts.json`. Apart from the mounts, you can also configure additional options like the Environment Variables and amount of Shared Memory available to the TAO launcher.

`IMPORTANT NOTE:` The code below creates a sample `~/.tao_mounts.json` file. Here, we can map directories in which we save the data, specs, results and cache. You should configure it for your specific case so these directories are correctly visible to the docker container.

In [ ]:

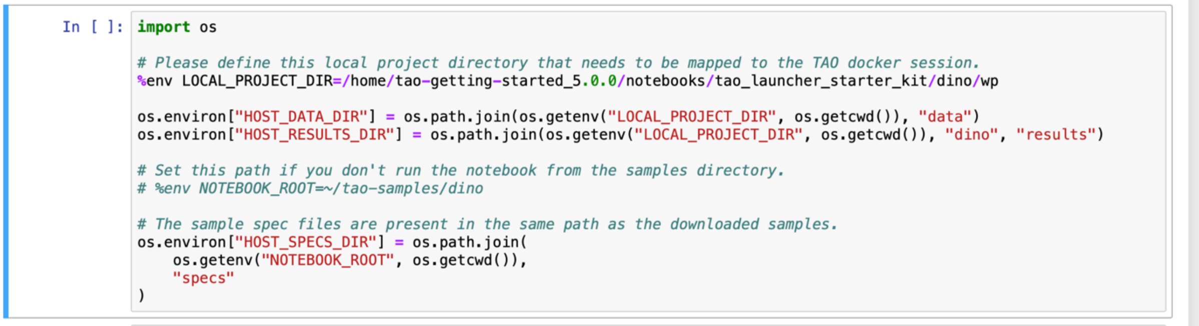
```
import os

# Please define this local project directory that needs to be mapped to the TAO docker session.
%env LOCAL_PROJECT_DIR=/path/to/local/tao-experiments

os.environ["HOST_DATA_DIR"] = os.path.join(os.getenv("LOCAL_PROJECT_DIR", os.getcwd()), "data")
os.environ["HOST_RESULTS_DIR"] = os.path.join(os.getenv("LOCAL_PROJECT_DIR", os.getcwd()), "dino", "results")

# Set this path if you don't run the notebook from the samples directory.
# %env NOTEBOOK_ROOT=~/tao-samples/dino

# The sample spec files are present in the same path as the downloaded samples.
os.environ["HOST_SPECS_DIR"] = os.path.join(
    os.getenv("NOTEBOOK_ROOT", os.getcwd()),
    "specs"
)
```



Select items to perform actions on them.

/ tao-getting-started_5.0.0 / notebooks / tao_launcher_starter_kit / dino

Name	Last Modified	File size
..	seconds ago	
specs	a month ago	
wp	2 days ago	
Crowd Counting.ipynb	a month ago	1.47 MB
dino-Copy1.ipynb	a month ago	27.9 kB
dino.ipynb	2 days ago	27.6 kB
sample.jpg	a year ago	107 kB



In [ ]:

```
import os

# Please define this local project directory that needs to be mapped to the TAO docker session.
%env LOCAL_PROJECT_DIR=/home/tao-getting-started_5.0.0/notebooks/tao_launcher_starter_kit/dino/wp

os.environ["HOST_DATA_DIR"] = os.path.join(os.getenv("LOCAL_PROJECT_DIR", os.getcwd()), "data")
os.environ["HOST_RESULTS_DIR"] = os.path.join(os.getenv("LOCAL_PROJECT_DIR", os.getcwd()), "dino", "results")

# Set this path if you don't run the notebook from the samples directory.
# %env NOTEBOOK_ROOT=~/tao-samples/dino

# The sample spec files are present in the same path as the downloaded samples.
os.environ["HOST_SPECS_DIR"] = os.path.join(
    os.getenv("NOTEBOOK_ROOT", os.getcwd()),
    "specs"
)
```

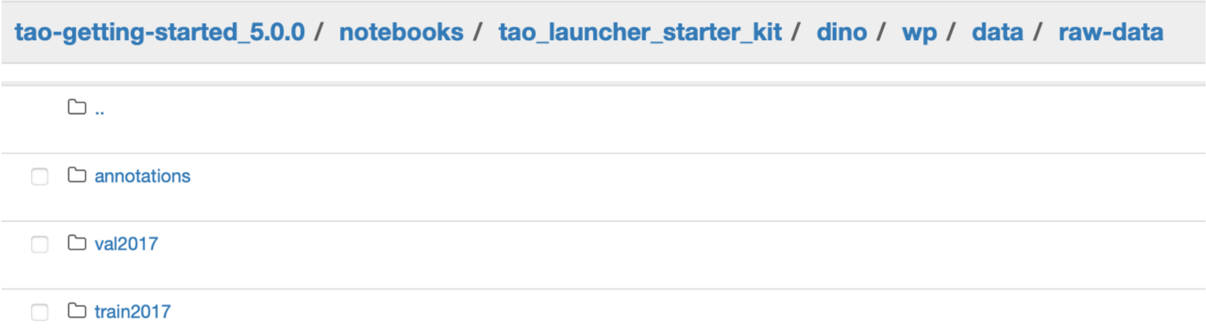
5. Upload and Unzip Dataset:

- Upload your dataset to the Jupyter notebook and unzip it.
- Unzip command:

```
!unzip /path/to/the-zip-file.zip -d /path/to/destination
```

- After unzipping, you will see two folders: train and valid.
- Move these two folders to the raw-data folder and rename them to train_2017 and val_2017, respectively.

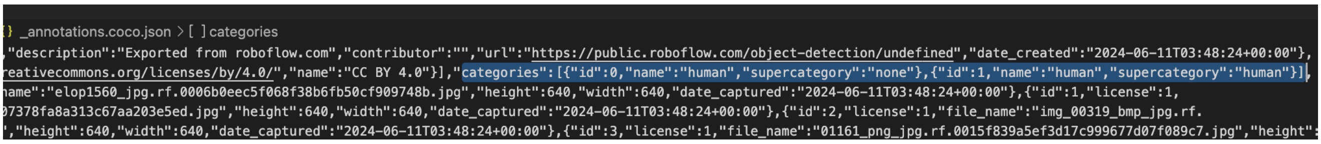
- Rename the _annotations.coco.json file in train_2017 to instances_train2017.json and in val_2017 to instances_val2017.json.
- Create a folder named annotations under the raw-data folder and move instances_train2017.json and instances_val2017.json to this folder.



tao-getting-started_5.0.0 / notebooks / tao_launcher_starter_kit / dino / wp / data / raw-data

- ..
- annotations
- val2017
- train2017

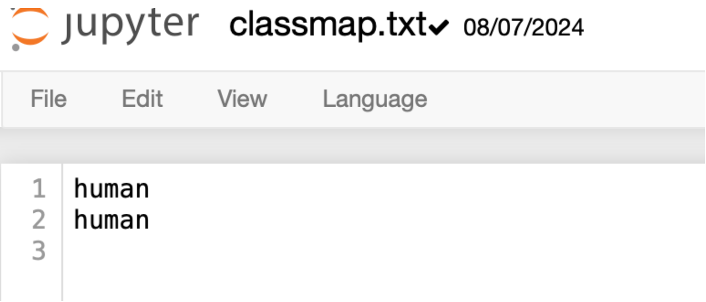
6. Configuration



{} _annotations.coco.json > [ ] categories

```
,"description":"Exported from roboflow.com","contributor":"","url":"https://public.roboflow.com/object-detection/undefined","date_created":"2024-06-11T03:48:24+00:00"},
reativecommons.org/licenses/by/4.0/","name":"CC BY 4.0"}],"categories":[{"id":0,"name":"human","supercategory":"none"},{"id":1,"name":"human","supercategory":"human"}],
name":"elop1560_jpg.rf.0006b0eec5f068f38b6fb50cf909748b.jpg","height":640,"width":640,"date_captured":"2024-06-11T03:48:24+00:00"},{"id":1,"license":1,
07378fa8a313c67aa203e5ed.jpg","height":640,"width":640,"date_captured":"2024-06-11T03:48:24+00:00"},{"id":2,"license":1,"file_name":"img_00319_bmp_jpg.rf.
","height":640,"width":640,"date_captured":"2024-06-11T03:48:24+00:00"},{"id":3,"license":1,"file_name":"01161_png_jpg.rf.0015f839a5ef3d17c999677d07f089c7.jpg","height":
```

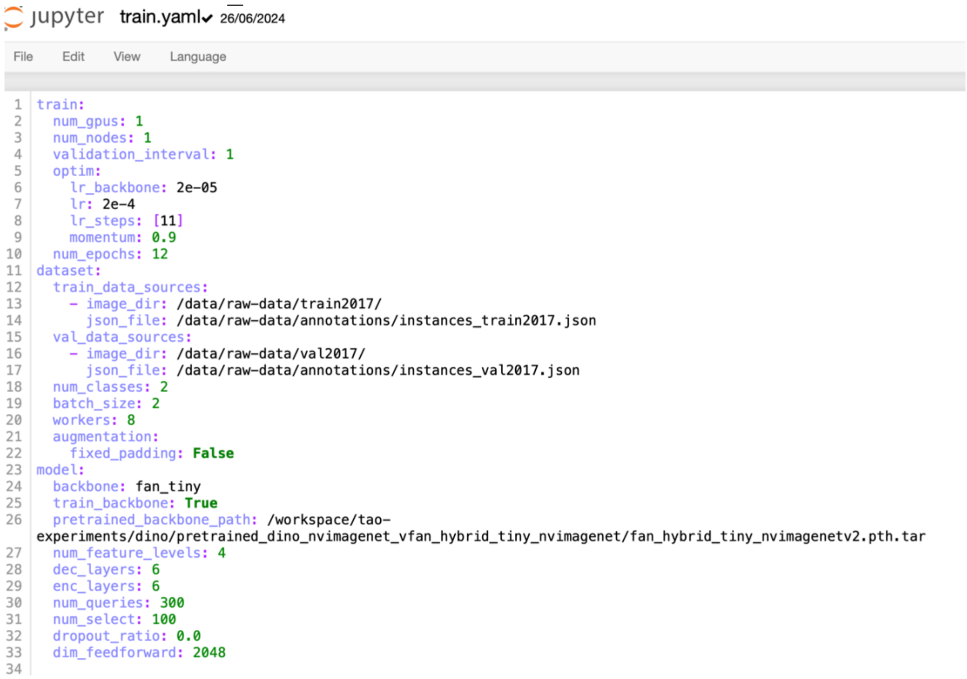
- **Configure Files:**
  - In the file `_annotations.coco.json`, ensure there are two classes: `human`.
  - Before starting the training model, configure some files in the `spec` folder:
    - `classmap.txt`: Specify the class names.



```
human
human

```

    - `train.yaml`:

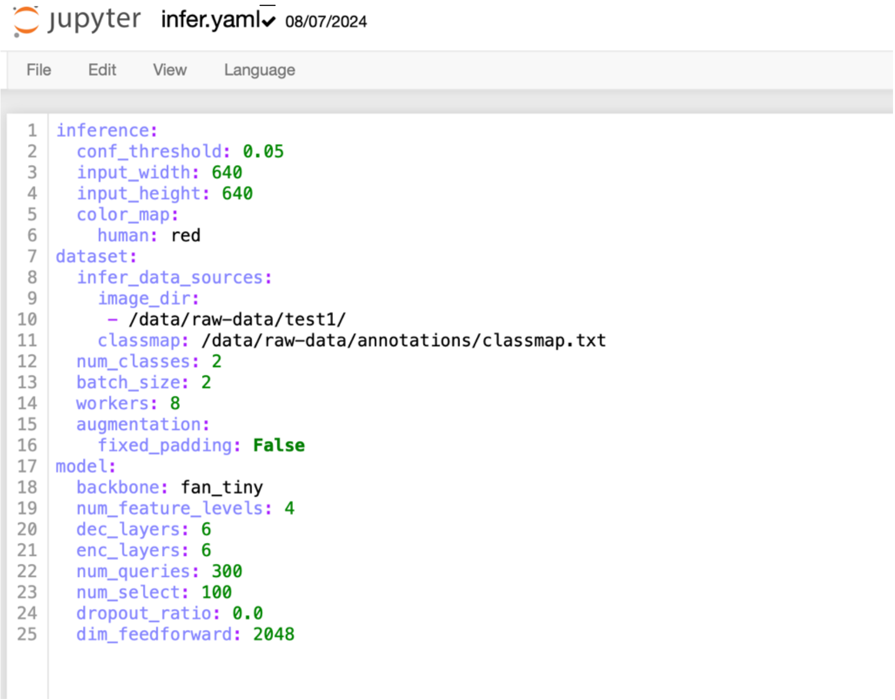
```
num_classes: 2
```



```yaml
train:
  num_gpus: 1
  num_nodes: 1
  validation_interval: 1
  optim:
    lr_backbone: 2e-05
    lr: 2e-4
    lr_steps: [11]
    momentum: 0.9
  num_epochs: 12
dataset:
  train_data_sources:
    - image_dir: /data/raw-data/train2017/
      json_file: /data/raw-data/annotations/instances_train2017.json
  val_data_sources:
    - image_dir: /data/raw-data/val2017/
      json_file: /data/raw-data/annotations/instances_val2017.json
  num_classes: 2
  batch_size: 2
  workers: 8
  augmentation:
    fixed_padding: False
model:
  backbone: fan_tiny
  train_backbone: True
  pretrained_backbone_path: /workspace/tao-experiments/dino/pretrained_dino_nvimagenet_vfan_hybrid_tiny_nvimagenet/fan_hybrid_tiny_nvimagenetv2.pth.tar
  num_feature_levels: 4
  dec_layers: 6
  enc_layers: 6
  num_queries: 300
  num_select: 100
  dropout_ratio: 0.0
  dim_feedforward: 2048

```

- **infer**.yaml:

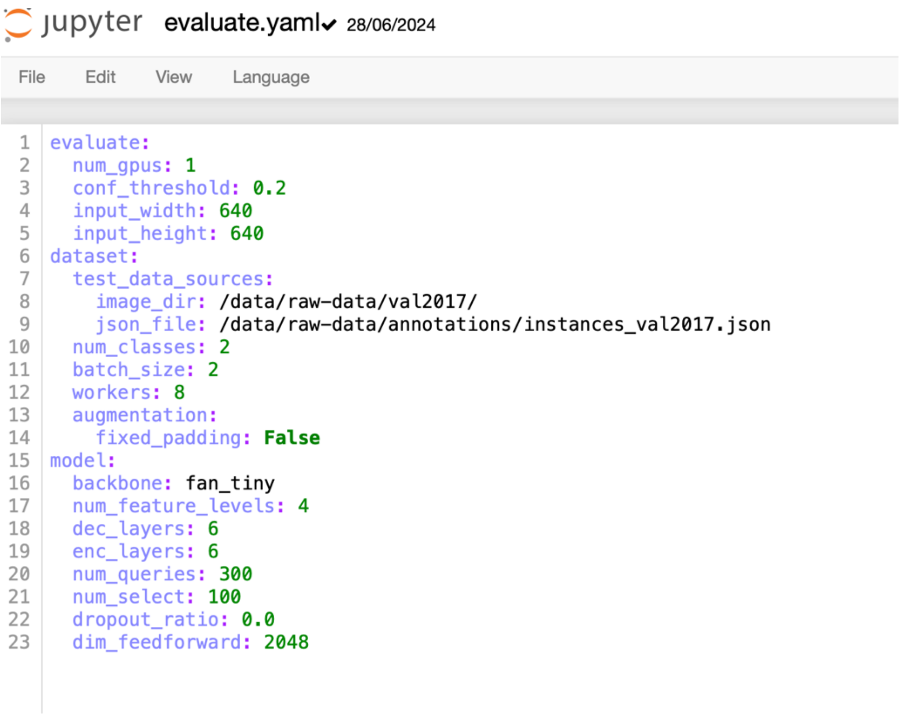
```
input_width: 640
input_height: 640
color_map:
      human:red
num_classes: 2
```



```yaml
inference:
  conf_threshold: 0.05
  input_width: 640
  input_height: 640
  color_map:
    human: red
dataset:
  infer_data_sources:
    image_dir:
      - /data/raw-data/test1/
    classmap: /data/raw-data/annotations/classmap.txt
  num_classes: 2
  batch_size: 2
  workers: 8
  augmentation:
    fixed_padding: False
model:
  backbone: fan_tiny
  num_feature_levels: 4
  dec_layers: 6
  enc_layers: 6
  num_queries: 300
  num_select: 100
  dropout_ratio: 0.0
  dim_feedforward: 2048
```

- **evaluate**.yaml:

```
input_width: 640
input_height: 640
num_classes: 2
```



```yaml
evaluate:
  num_gpus: 1
  conf_threshold: 0.2
  input_width: 640
  input_height: 640
dataset:
  test_data_sources:
    image_dir: /data/raw-data/val2017/
    json_file: /data/raw-data/annotations/instances_val2017.json
  num_classes: 2
  batch_size: 2
  workers: 8
  augmentation:
    fixed_padding: False
model:
  backbone: fan_tiny
  num_feature_levels: 4
  dec_layers: 6
  enc_layers: 6
  num_queries: 300
  num_select: 100
  dropout_ratio: 0.0
  dim_feedforward: 2048
```

7. Start Training the Model

- Follow the instructions in the **dino.ipynb** to start training the model.

Table of Contents

This notebook shows an example usecase of DINO using Train Adapt Optimize (TAO) Toolkit.

0. Set up env variables and map drives
1. Installing the TAO launcher
2. Prepare dataset and pre-trained model
3. Provide training specification
4. Run TAO training
5. Evaluate a trained model
6. Visualize inferences
7. Deploy