MASK RCNN Using Tensorflow OBJECT DETECTION API

Name: kishor G

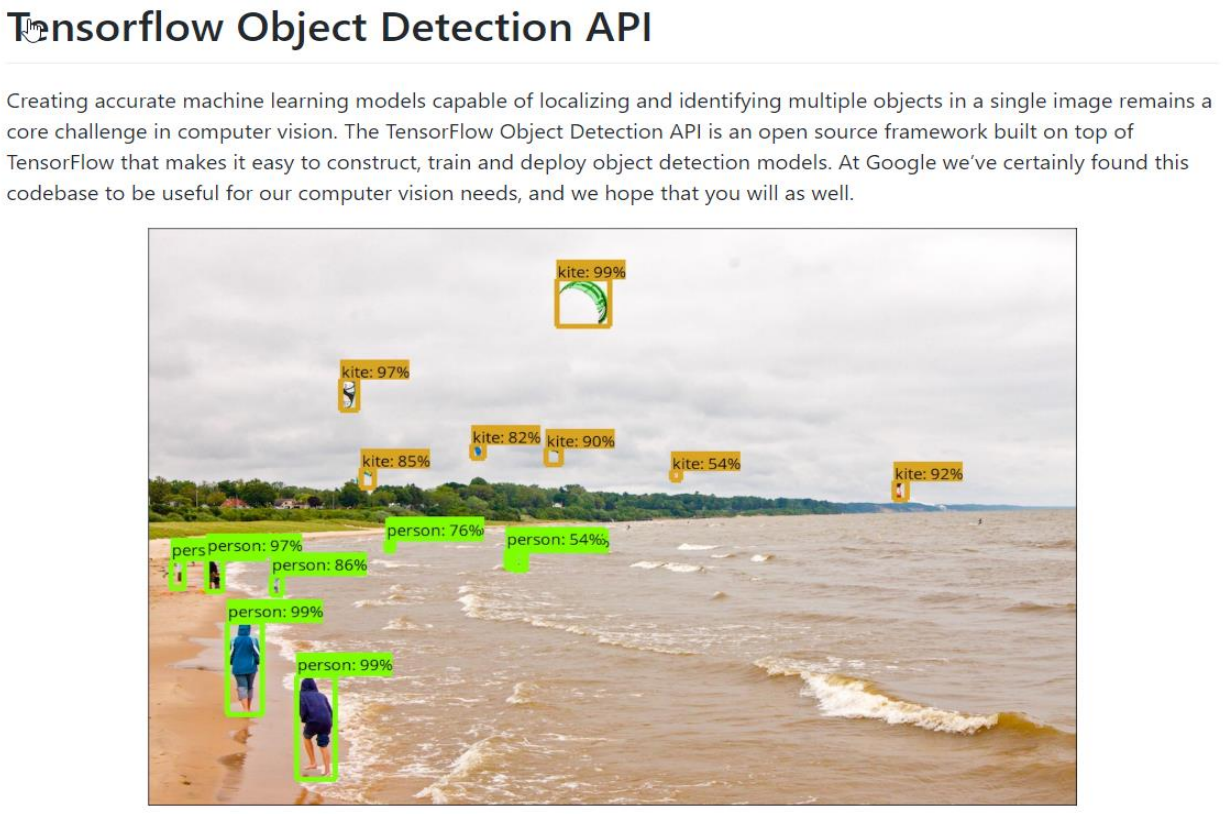
Email: kishorbrindha18@gmail.com

Youtube_link: https://youtu.be/YMlbFNFnzII

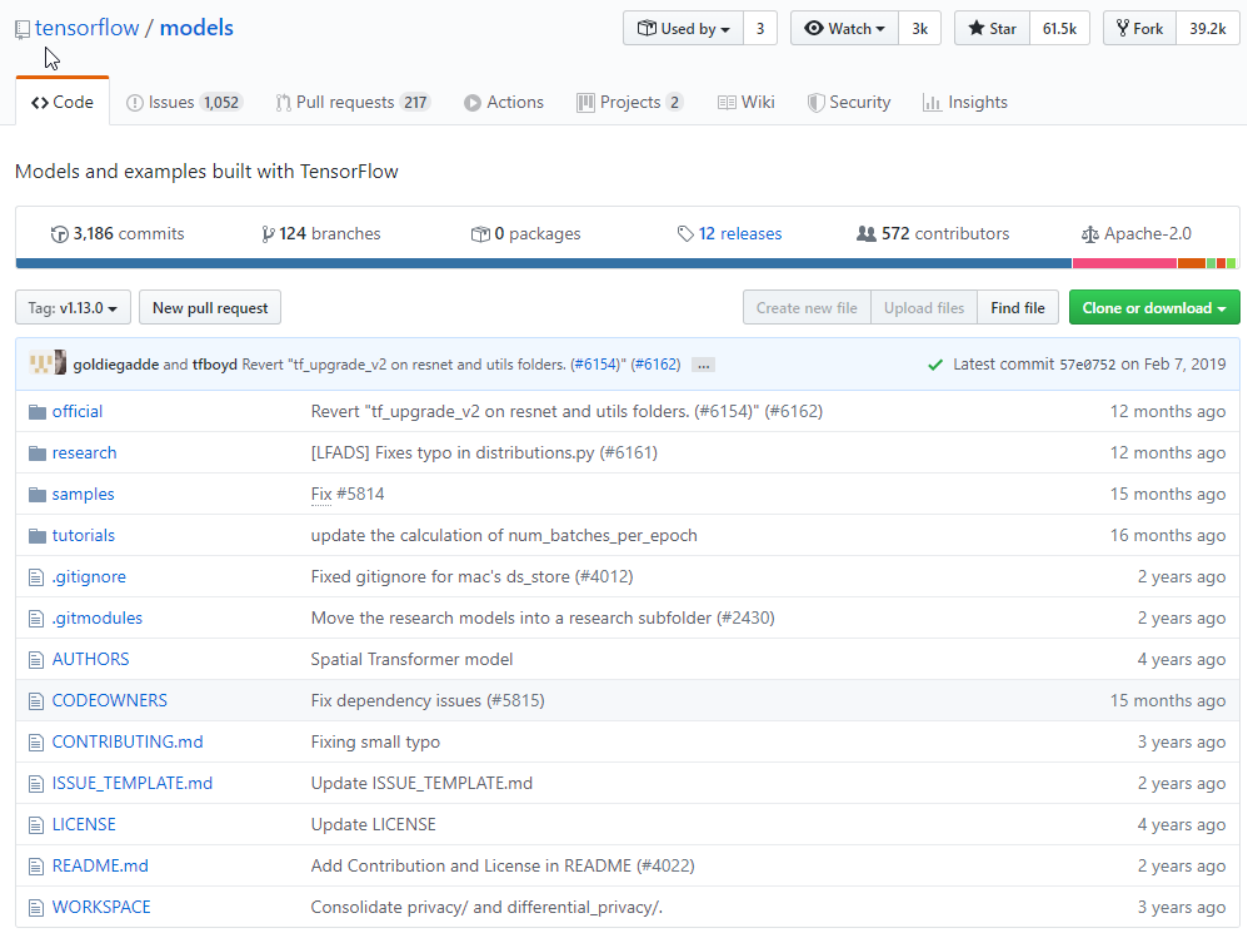
Github_link: https://github.com/kishorsumathi/mask_rcnn_codebugged_AI

In June 2017, Google opened the TensorFlow Object Detection API. This project uses TensorFlow to implement most of the deep learning target detection frameworks, including MaskRCNN in reality.

https://github.com/tensorflow/models/tree/master/research/object_detection



https://github.com/tensorflow/models/tree/v1.13.0



tensorflow / models

Used by 3 | Watch 3k | Star 61.5k | Fork 39.2k

Code | Issues 1,052 | Pull requests 217 | Actions | Projects 2 | Wiki | Security | Insights

Models and examples built with TensorFlow

3,186 commits | 124 branches | 0 packages | 12 releases | 572 contributors | Apache-2.0

Tag: v1.13.0 | New pull request | Create new file | Upload files | Find file | Clone or download

goldiegadde and tfboyd Revert "tf_upgrade_v2 on resnet and utils folders. (#6154)" (#6162) ... Latest commit 57e0752 on Feb 7, 2019

official	Revert "tf_upgrade_v2 on resnet and utils folders. (#6154)" (#6162)	12 months ago
research	[LFADS] Fixes typo in distributions.py (#6161)	12 months ago
samples	Fix #5814	15 months ago
tutorials	update the calculation of num_batches_per_epoch	16 months ago
.gitignore	Fixed gitignore for mac's ds_store (#4012)	2 years ago
.gitmodules	Move the research models into a research subfolder (#2430)	2 years ago
AUTHORS	Spatial Transformer model	4 years ago
CODEOWNERS	Fix dependency issues (#5815)	15 months ago
CONTRIBUTING.md	Fixing small typo	3 years ago
ISSUE_TEMPLATE.md	Update ISSUE_TEMPLATE.md	2 years ago
LICENSE	Update LICENSE	4 years ago
README.md	Add Contribution and License in README (#4022)	2 years ago
WORKSPACE	Consolidate privacy/ and differential_privacy/.	3 years ago

Tensorflow detection model zoo

We provide a collection of detection models pre-trained on the COCO dataset, the Kitti dataset, the Open Images dataset, the AVA v2.1 dataset and the iNaturalist Species Detection Dataset. These models can be useful for out-of-the-box inference if you are interested in categories already in those datasets. They are also useful for initializing your models when training on novel datasets.

In the table below, we list each such pre-trained model including:

- a model name that corresponds to a config file that was used to train this model in the `samples/configs` directory,
- a download link to a tar.gz file containing the pre-trained model,
- model speed --- we report running time in ms per 600x600 image (including all pre and post-processing), but please be aware that these timings depend highly on one's specific hardware configuration (these timings were performed using an Nvidia GeForce GTX TITAN X card) and should be treated more as relative timings in many cases. Also note that desktop GPU timing does not always reflect mobile run time. For example Mobilenet V2 is faster on mobile devices than Mobilenet V1, but is slightly slower on desktop GPU.
- detector performance on subset of the COCO validation set or Open Images test split as measured by the dataset-specific mAP measure. Here, higher is better, and we only report bounding box mAP rounded to the nearest integer.
- Output types (`Boxes` , and `Masks` if applicable)

MASK RCNN Model link :-

http://download.tensorflow.org/models/object_detection/mask_rcnn_inception_v2_coco_2018_01_28.tar.gz

mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks
mask_rcnn_inception_v2_coco	79	25	Masks
mask_rcnn_resnet101_atrous_coco	470	33	Masks
mask_rcnn_resnet50_atrous_coco	343	29	Masks

- **Data Annotation:**

  In data annotation, we will be using **Labelme Tool**

  Download the Tool from the given Link:-

  https://github.com/wkentaro/labelme

  Installation of Labelme:

  conda create --name=labelme python=3.6

  conda activate labelme

  # conda install -c conda-forge pyside2

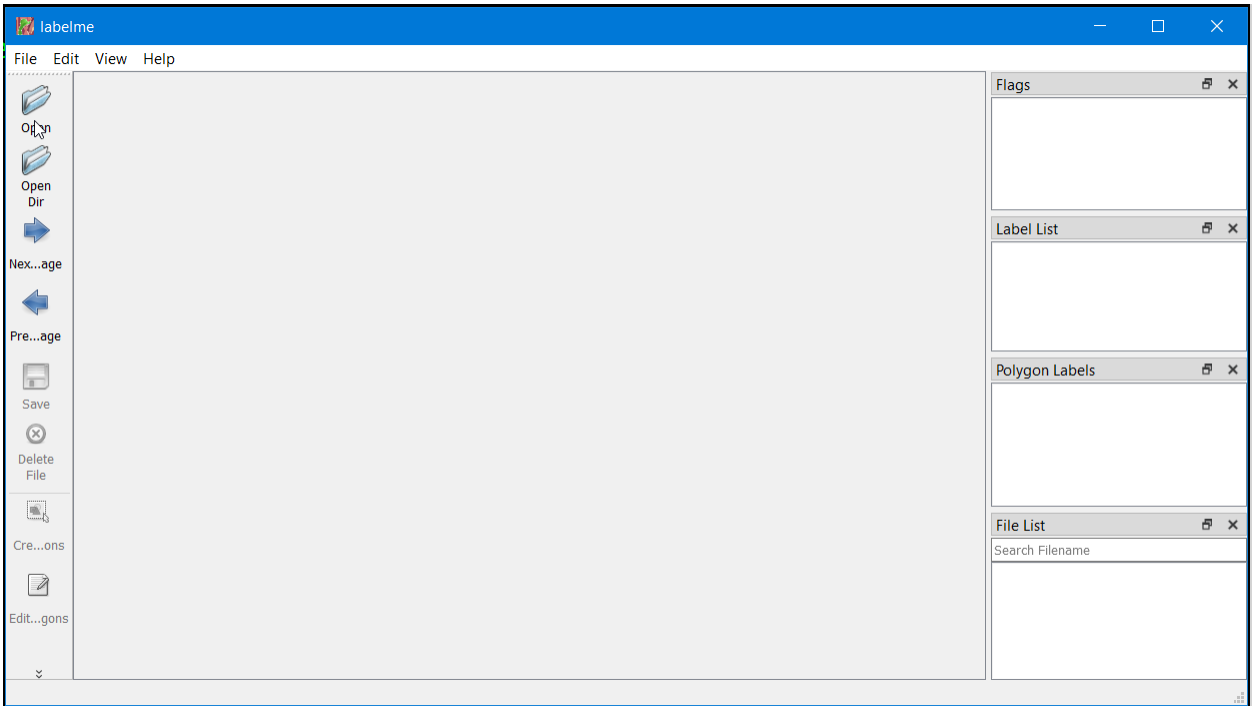
  # conda install pyqt

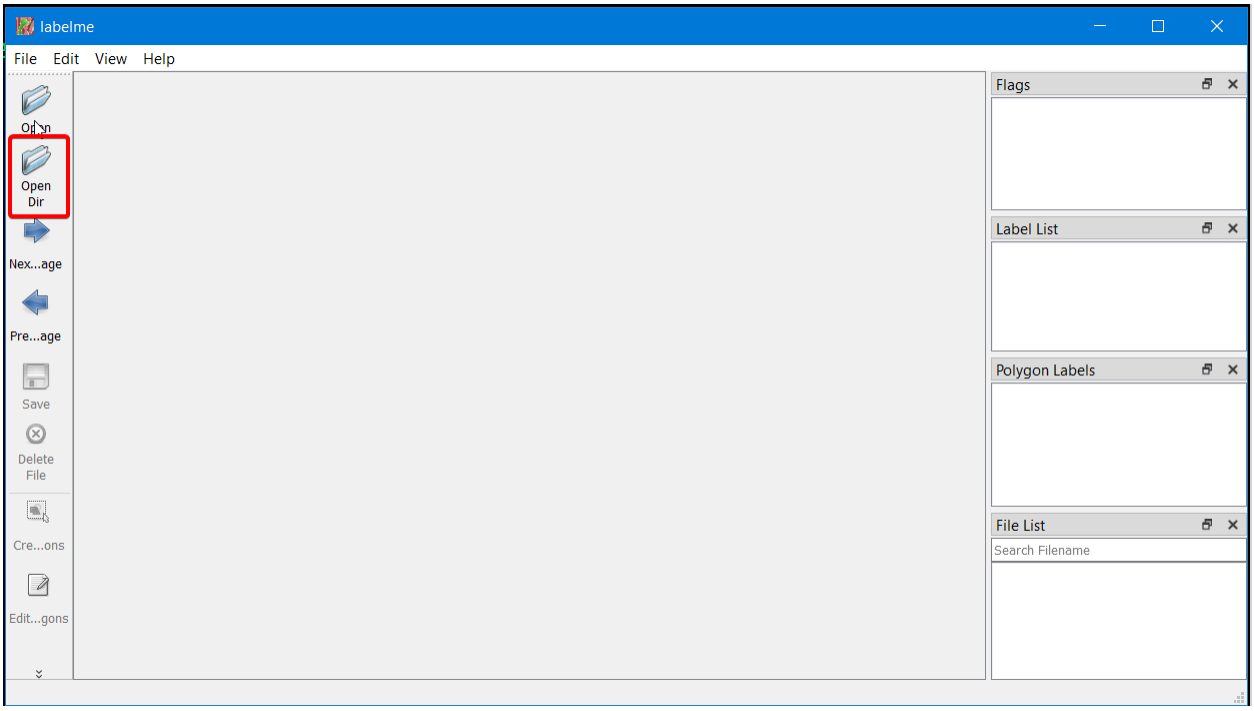
  # pip install pyqt5 # pyqt5 can be installed via pip on python3

  pip install labelme

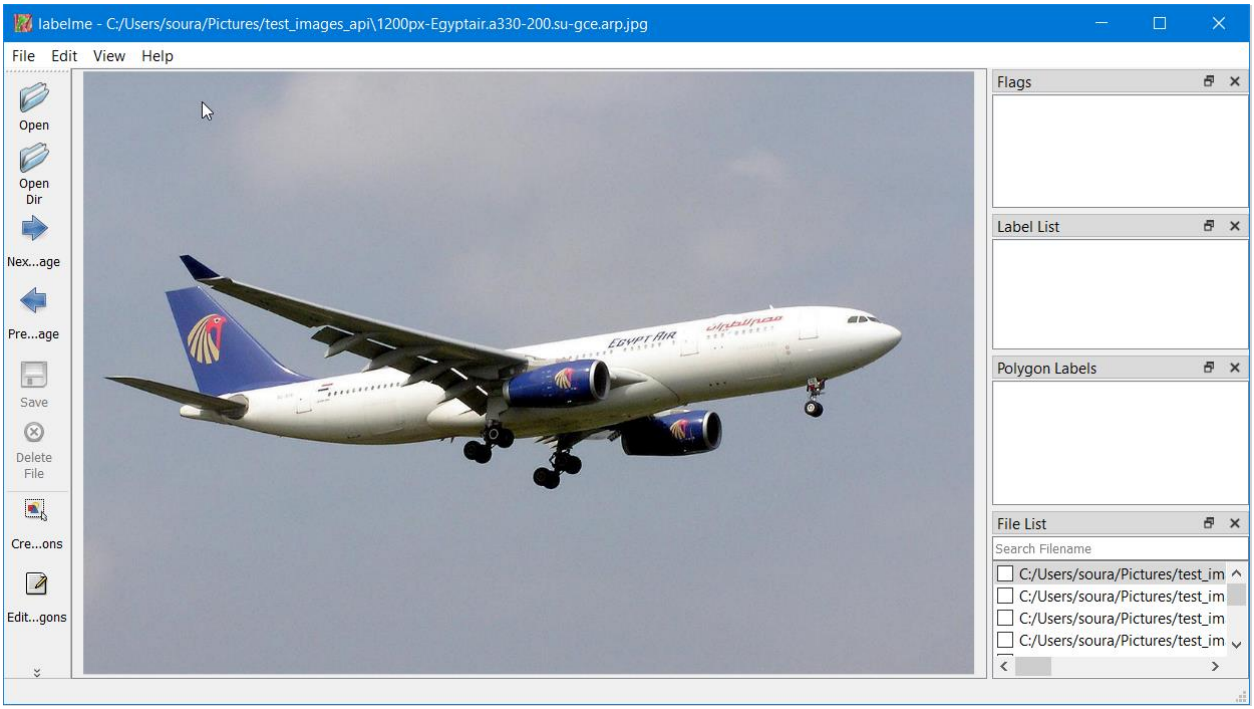
  labelme

Then the interface of labelme will popup.

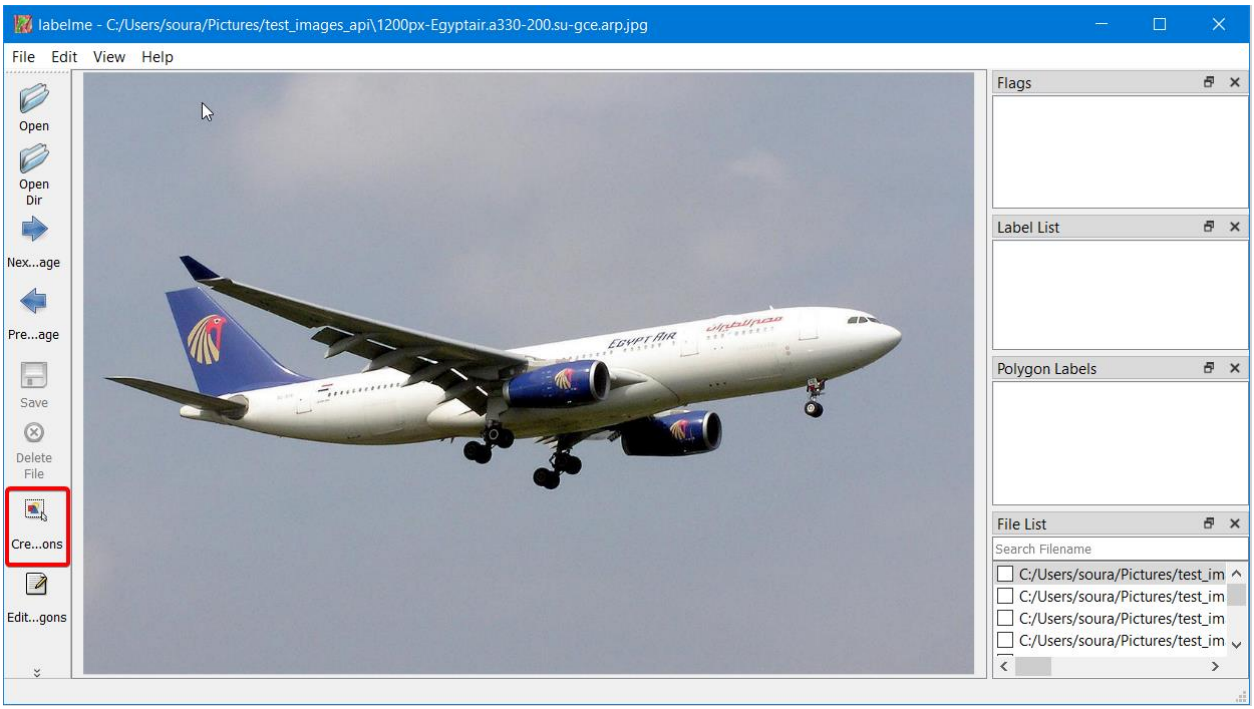




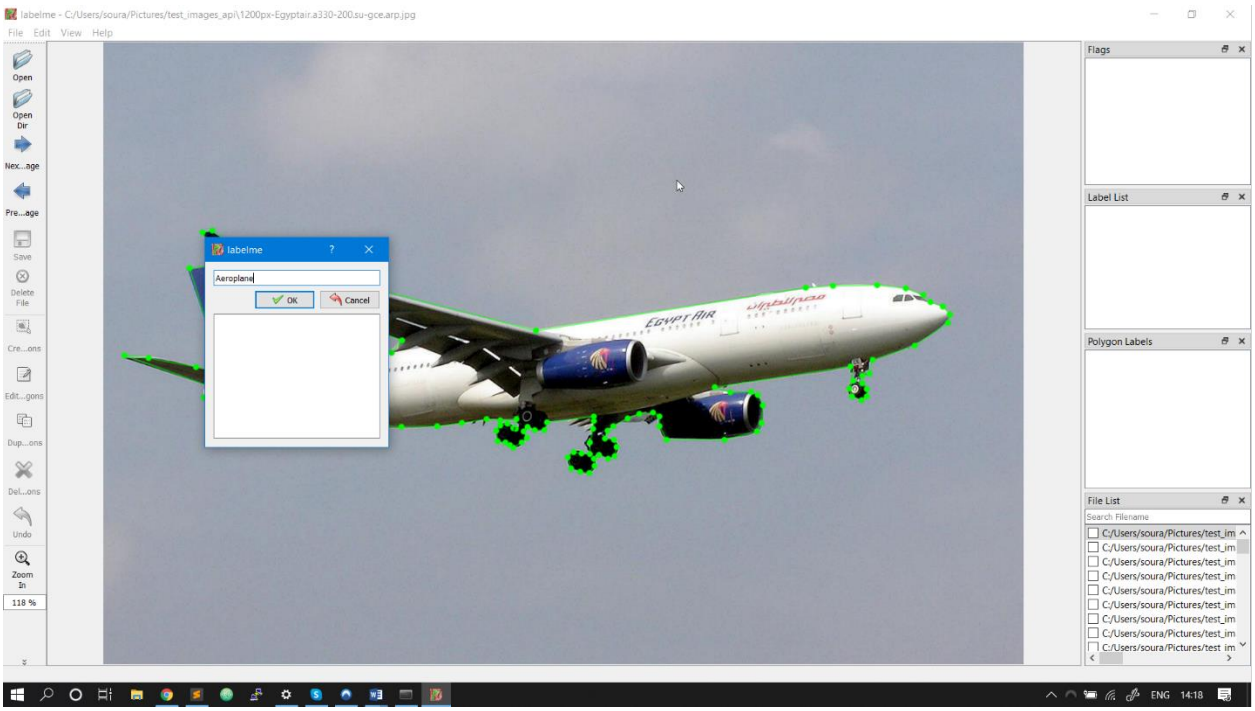
Click on **Open Dir** and select the folder where your test images or train images are. Then start labelling all images in the dataset.



The selected image is loaded in the Labelme Interface.

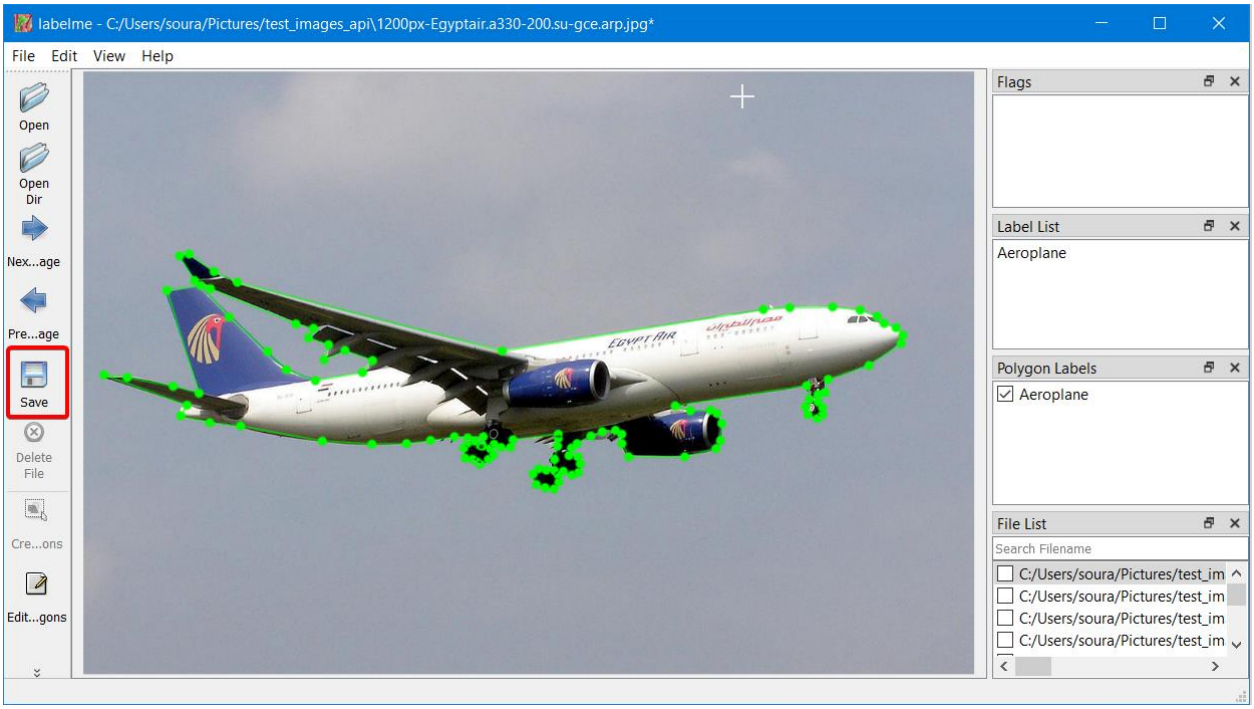


Click on **Start Drawing Polygons.**

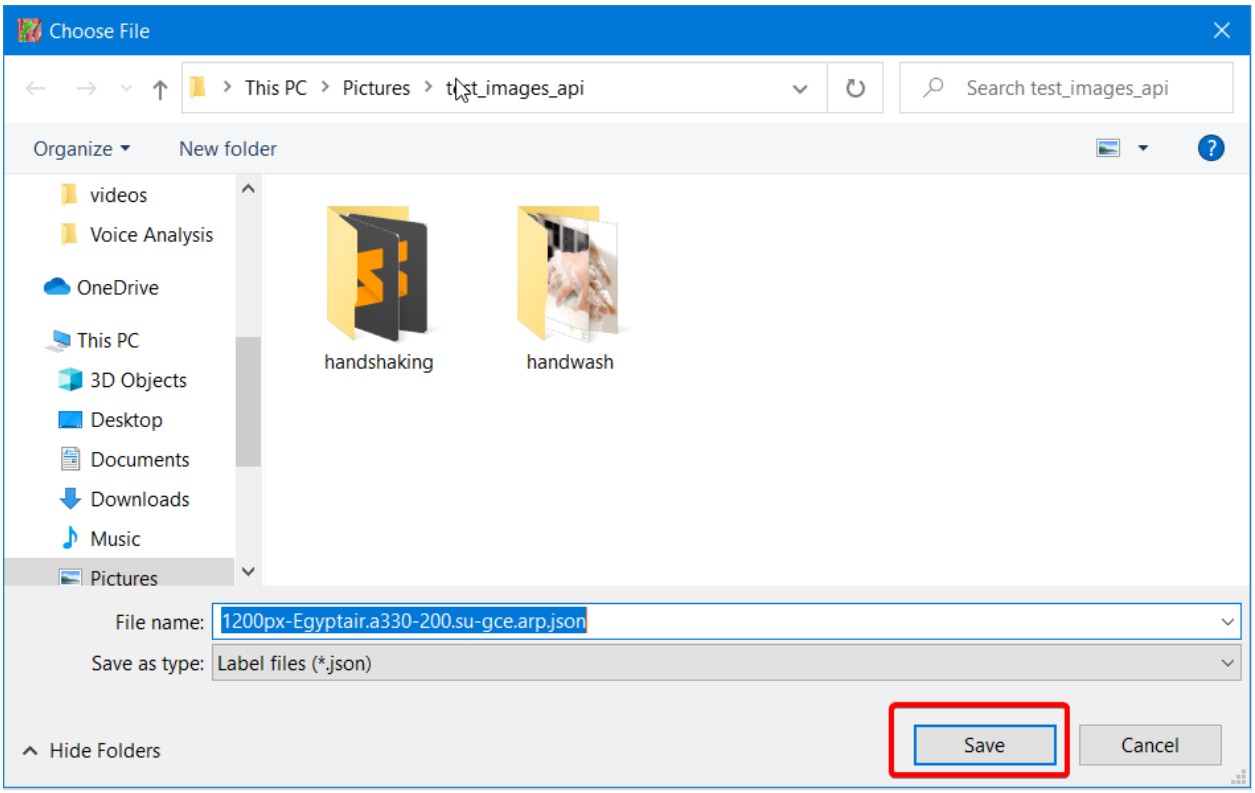


Create the Polygon Box and write the Label name corresponding to the image

.



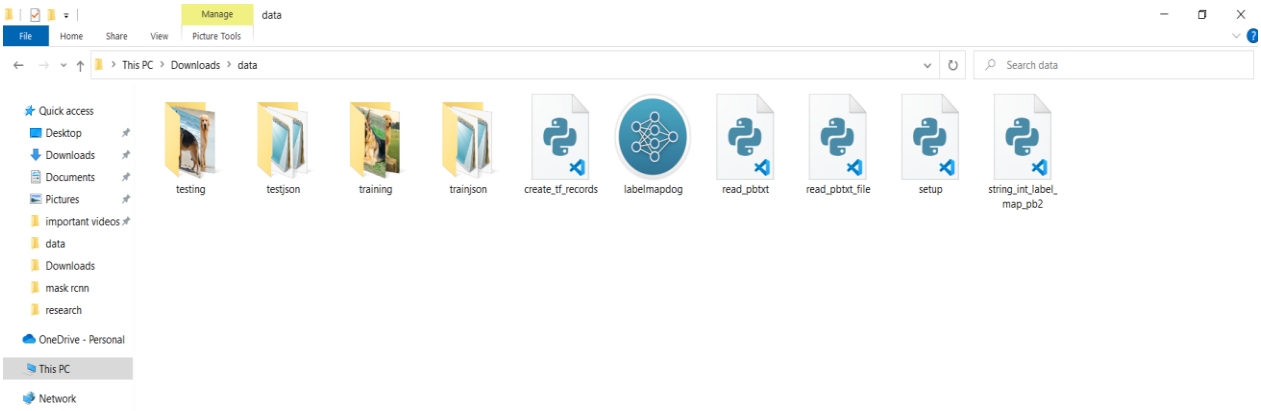
Click on the Save button and a corresponding JSON file will be saved in the directory with the image.



NEXT

Split the dataset into 80:20 ratio for train and validation(use splitfolders Library)

Create a new folder named Data in your local system. This is where we will be keeping our own trainingdata,testdata,trainjson(annotated),testjson(annotated),label map.pbtxt,read_pbtxt,read_pbtxt_file,setup.py,string_int_label_map_pb2 and create_tf_records.



create_tf_records:

As we are using TFOD framework for training maskrcnn you have to convert all the trainjson and testjson annotated images to tf_records. So we use create_tf_record.py file and we will perform the conversion.
read_pbtxt ,read_pbtxt_file,string_int_label_map_pb2 are helper file for **c**reate_tf_records.py file

Setup.py:

As we are following modular coding approach we create a setup.py file and install it to make object_detection as local package.

labelmap.pbtxt:

The label map tells the trainer what each object is by defining a mapping of class names to class ID numbers. Use a text editor to create a new file and save it as labelmap.pbtxt

Replace the names of the classes with your class names.

This is a sample one.

```
item {
    id: 1
    name: 'Goggles'
}
item {
    id: 2
    name: 'Hat'
}
item {
    id: 3
    name: 'Jacket'
}
item {
    id: 4
    name: 'Shirt'
}
item {
    id: 5
    name: 'Shorts'
}
item {
    id: 6
    name: 'Trouser'
}
item {
    id: 7
    name: 'T-Shirt'
}
item {
    id: 8
    name: 'Wallet'
}
item {
    id: 9
    name: 'Watch'
}
item {
    id: 10
    name: 'Shoes'
}
```

Zip the data folder and keep it ready we will be using this folder later.

Colab_file_link:

https://colab.research.google.com/drive/1NmyMeMqmGT2uDHIPjh6WnC6MYK4XHY4e?usp=sharing

- **GOOGLE_COLAB_SETUP:**
    - **Step1: Mount your Google drive**
    - **Step 2: Download the tensorflow repository**
    - **Step 3: Download the mask rcnn pretrained model and unzip it**

- **Step 4: Install necessary requirements**

- **Step 5: upload data and other requirements in zip file format(data.zip)**

- **step 6: convert .json to .tf-records and create tf-records**

  Changes in create_tf_records.py for both train and test Data:

  - ✓ trainImagePath(index-246)
  - ✓ trainImagePath(json)
  - ✓ labelMapPath
  - ✓ outputFolderPath(index-249)

  Change the path for both training and test data save and close the file.

Train data:

```
#trainImageJsonPath = "../research/object_detection/mycustomdataset/train_jsons"
#labelMapPath = "../research/object_detection/data/labelmapcustom.pbtxt"
#outputFolderPath = "../research/object_detection/data/custom_train.record"

trainImagePath = "/content/mask_rcnn_training/data/training"
trainImageJsonPath = "/content/mask_rcnn_training/data/trainjson"
labelMapPath = "/content/mask_rcnn_training/data/labelmapdog.pbtxt"
outputFolderPath = "/content/mask_rcnn_training/data/custom_train_dog.record"


# label_map = read_pbtxt_file.get_label_map_dict(FLAGS.label_map_path)
label_map = read_pbtxt_file.get_label_map_dict(labelMapPath)
writer = tf.python_io.TFRecordWriter(outputFolderPath)
annotations_json_path = os.path.join(trainImageJsonPath, '*.json')
# writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
```

Test data:

```
#trainImageJsonPath = "../research/object_detection/mycustomdataset/train_jsons"
#labelMapPath = "../research/object_detection/data/labelmapcustom.pbtxt"
#outputFolderPath = "../research/object_detection/data/custom_train.record"

trainImagePath = "/content/mask_rcnn_training/data/testing"
trainImageJsonPath = "/content/mask_rcnn_training/data/testjson"
labelMapPath = "/content/mask_rcnn_training/data/labelmapdog.pbtxt"
outputFolderPath = "/content/mask_rcnn_training/data/custom_val_dog.record"



# label_map = read_pbtxt_file.get_label_map_dict(FLAGS.label_map_path)
label_map = read_pbtxt_file.get_label_map_dict(labelMapPath)
writer = tf.python_io.TFRecordWriter(outputFolderPath)
annotations_json_path = os.path.join(trainImageJsonPath, '*.json')
# writer = tf.python_io.TFRecordWriter(FLAGS.output_path)

num_annotations_skiped = 0
# annotations_json_path = os.path.join(FLAGS.annotations_json_dir, '*.json')
for i, annotation_file in enumerate(glob.glob(annotations_json_path)):
    if i % 100 == 0:
        print('On image %d', i)
```

- **Step 7: move the custom_train_dog.record ,custom_val_dog.record and setup.py to research folder. And install setup.py.**

- **Step 8: create a folder named training in research folder and move frozen_inference_graph.pb from mask_rcnn(pretrained_model) folder to training folder,move labelmap.pbtxt from data folder to training folder. And copy the file named mask_rcnn_inception_v2_coco.config from the given path /content/mask_rcnn_training/models/research/object_detection/samples/configs/mask_rcnn_inception_v2_coco.config and paste it in training folder**

- **Step 9: changes to be made in mask_rcnn_inception_v2_coco.config**

Changes:

- ✓ 10->num_classes-1
- ✓ 127->fine_tune_checkpoint- "mask_rcnn/model.ckpt"
- ✓ 133->num_steps- 200
- ✓ 142->input_path- "custom_train_dog.record"
- ✓ 144->label_map_path- "training/labelmapdog.pbtxt"
- ✓ 158->input path- "custom_val_dog.record".
- ✓ 160->label_map_path- "training/labelmapdog.pbtxt"

Make the above changes save and close the file

```
model {
  faster_rcnn {
    num_classes: 11
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_inception_v2'
      first_stage_features_stride: 16
    }
```

```
gradient_clipping_by_norm: 10.0
fine_tune_checkpoint: "maskrcnn/model.ckpt"
from_detection_checkpoint: true
# Note: The below line limits the training process to 200K steps, which we
# empirically found to be sufficient enough to train the pets dataset. This
# effectively bypasses the learning rate schedule (the learning rate will
# never decay). Remove the below line to train indefinitely.
num_steps: 200000
data_augmentation_options {
  random_horizontal_flip {
```

```
  num_steps: 200000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}

train_input_reader: {
  tf_record_input_reader {
    input_path: "C:/tensorflow_object_detection/models/research/train.record"
  }
  label_map_path: "C:/tensorflow_object_detection/models/research/training/label_map.pbtxt"
}

eval_config: {
  num_examples: 8000
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  max_evals: 10
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "C:/tensorflow_object_detection/models/research/test.record"
  }
  label_map_path: "C:/tensorflow_object_detection/models/research/training/label_map.pbtxt"
  shuffle: false
  num_readers: 1
}
```

All the red box cells contain the path that I am using. It may change based on your preferences.

- **Step 10: copy train.py from object_detection/legacy and paste the file in research folder and start the training**

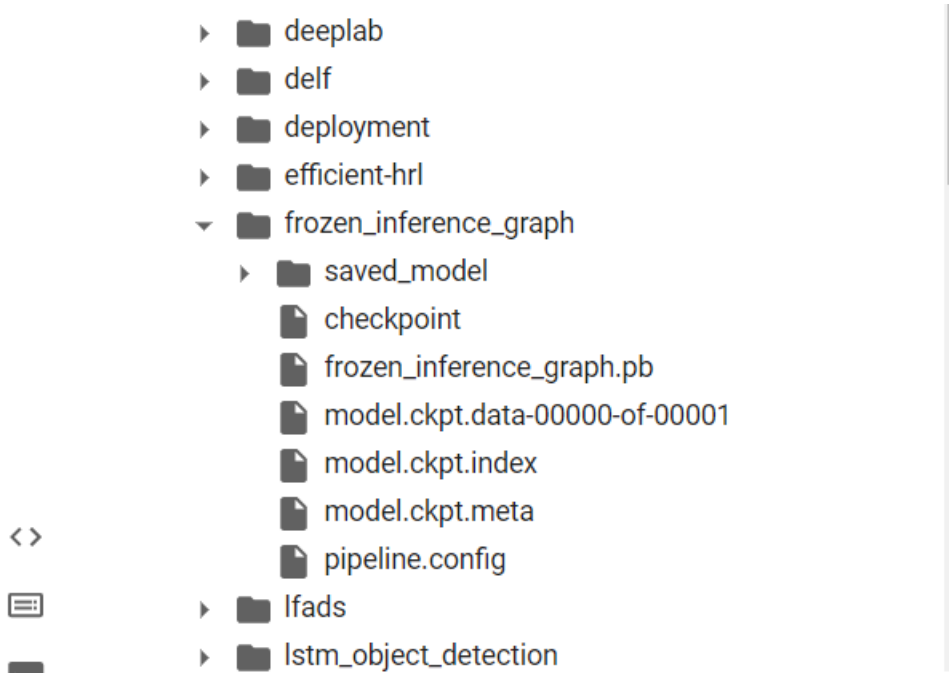
  - ✓ Start the training
  - ✓ Command for training:
    !python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/mask_rcnn_inception_v2_coco.config

- **Step 11: ckpt to pb conversion**

  - ✓ See the training/model.ckpt and give the latest ckpt number in the command for the conversion.
  - ✓ Command for conversion:
    !python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/mask_rcnn_inception_v2_coco.config --

trained_checkpoint_prefix training/model.ckpt-200 --output_directory frozen_inference_graph

✓ Training has completed successfully and model has been saved in research/frozen_inference_graph/ utilize that model for inferencing.



Now we can use this frozen_inference_graph.pb model to predict.

- **Now lets predict : -**

  - **Step 1: Necessary imports**
  - **Step 2: Model preparation**

    Changes:

    - ✓ PATH_TO_FROZEN_GRAPH
    - ✓ PATH_TO_LABLES

```
# Path to frozen detection graph. This is the actual model that is used for the object detection.
PATH_TO_FROZEN_GRAPH = "/content/mask_rcnn_training/models/research/frozen_inference_graph/frozen_inference_graph.pb"

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('training', 'labelmapdog.pbtxt')
```

  - **Step 4: Load a (frozen) Tensorflow model into memory.**
  - **Step 5: Loading label map.**
  - **Step 6: Helper code.**
  - **Step 7: upload test images and unzip it (test_image.zip).**
  - **Step 8:Detection**

Test data must be named as image1,image2,image3…..etc. This is done to access the images easily.

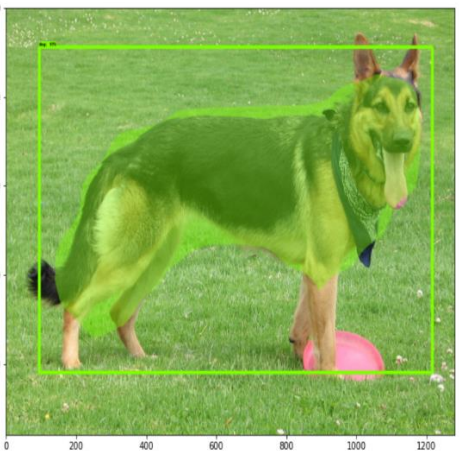
```
PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(1, 11) ]

# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)
```

Only you have to change the for loop if 5 images then range(1,6) if 20 image range(1,21) etc. And change the PATH_TO_IMAGES_DIR relative to your path were the test images is available.

Now execute all the remaining cells of the notebook and your test pictures will be displayed at the last cell.

RESULT:

```
    instance_masks=output_dict.get('detection_masks'),
    use_normalized_coordinates=True,
    line_thickness=8)
plt.figure(figsize=IMAGE_SIZE)
plt.imshow(image_np)
```



CONCLUSION:

I have trained the maskrcnn model with dog dataset for only 600 steps and I have got a pretty good result. If you increase the steps and dataset size your accuracy will increase**. Data argumentation and data annotation must be done very carefully.**