Ary Naim

anaim@unm.edu

Step by Step guide for Collab Notebook “PUBLIC_yolov8_fire_and_smoke_retrain.ipynb”

Complete training pipeline to train YOLOv8 classifier for fire & smoke detection based on images.

Prerequisite Steps:

1. You need to have a Gmail account, to upload your custom datasets to Google Drive.
2. Create a Google Collab account.

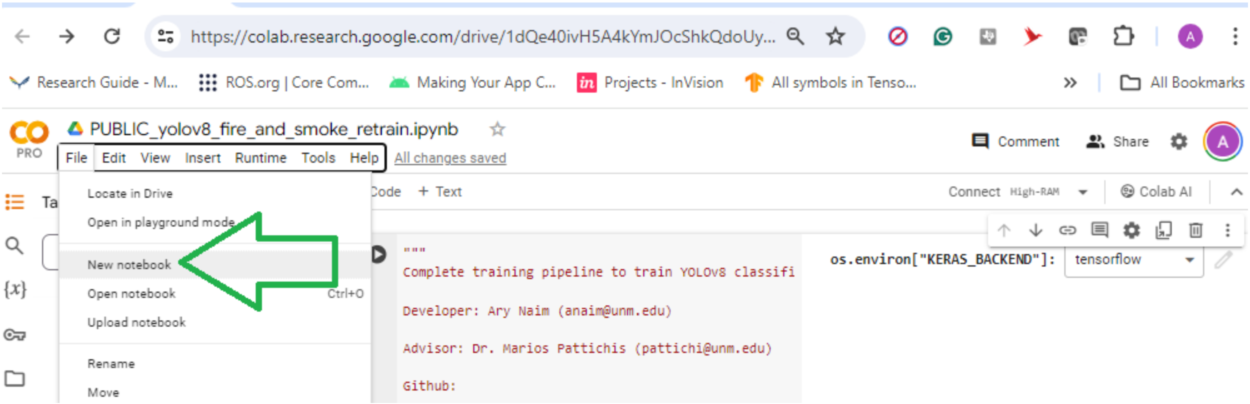
   This is an online programming environment that gives you access to GPUs, and CPUs, and can be shared with other people.

   Note: Google is not free after the 1st several hours and has both monthly costs and hourly costs for GPUs.

3. Navigate to “PUBLIC_yolov8_fire_and_smoke_retrain.ipynb” at

   https://colab.research.google.com/drive/1dQe40ivH5A4kYmJOcShkQdoUyI0sH5F0?usp=sharing

   This notebook can only be viewed or read therefore to make modifications create your Notebook in Collab.



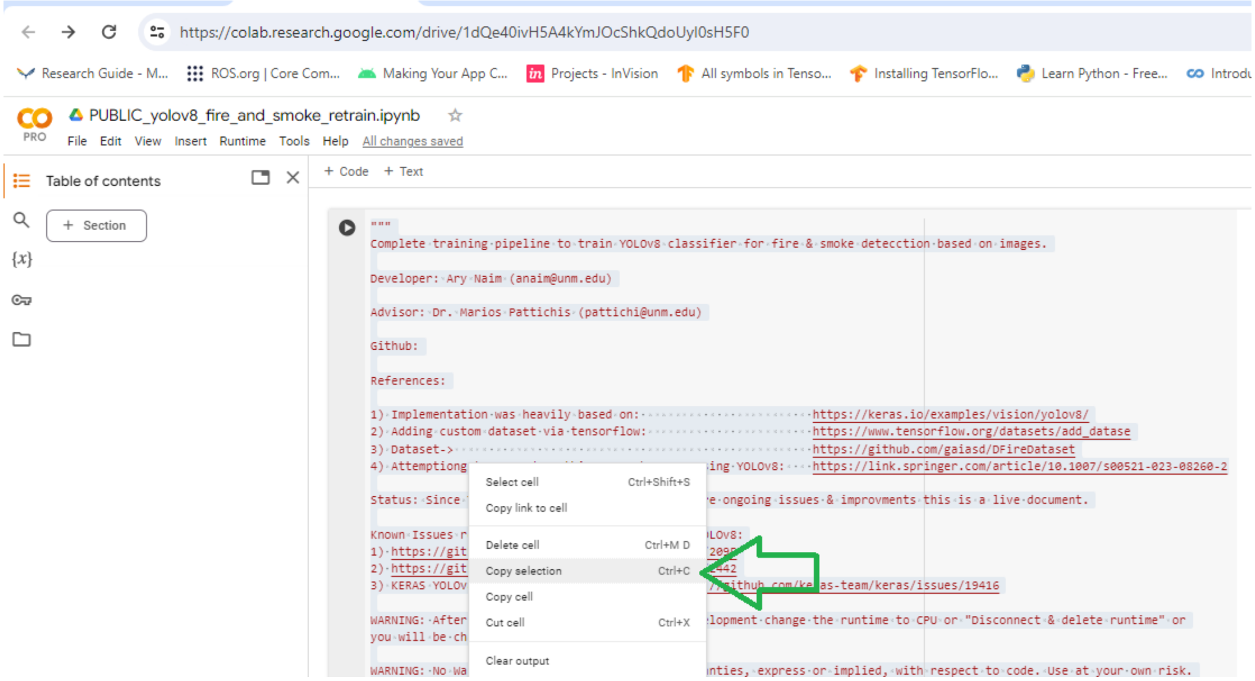
```
"""
Complete training pipeline to train YOLOv8 classifi

Developer: Ary Naim (anaim@unm.edu)

Advisor: Dr. Marios Pattichis (pattichi@unm.edu)

Github:
```

os.environ["KERAS_BACKEND"]: tensorflow

Let’s start:

Step 0:

Copy all of the code from “PUBLIC_yolov8_fire_and_smoke_retrain.ipynb” into your notebook.



```
"""
Complete training pipeline to train YOLOv8 classifier for fire & smoke detecction based on images.

Developer: Ary Naim (anaim@unm.edu)

Advisor: Dr. Marios Pattichis (pattichi@unm.edu)

Github:

References:

1) Implementation was heavily based on:                         https://keras.io/examples/vision/yolov8/
2) Adding custom dataset via tensorflow:                        https://www.tensorflow.org/datasets/add_datase
3) Dataset->                                                    https://github.com/gaiasd/DFireDataset
4) Attemptiong                                   ing YOLOv8:    https://link.springer.com/article/10.1007/s00521-023-08260-2

Status: Since                                    e ongoing issues & improvments this is a live document.

Known Issues r                                   LOv8:
1) https://git                                   209
2) https://git                                   442
3) KERAS YOLOv                                   //github.com/keras-team/keras/issues/19416

WARNING: After                                   lopment change the runtime to CPU or "Disconnect & delete runtime" or
you will be ch

WARNING: No Wa                                   nties, express or implied, with respect to code. Use at your own risk.
```

Step 1:

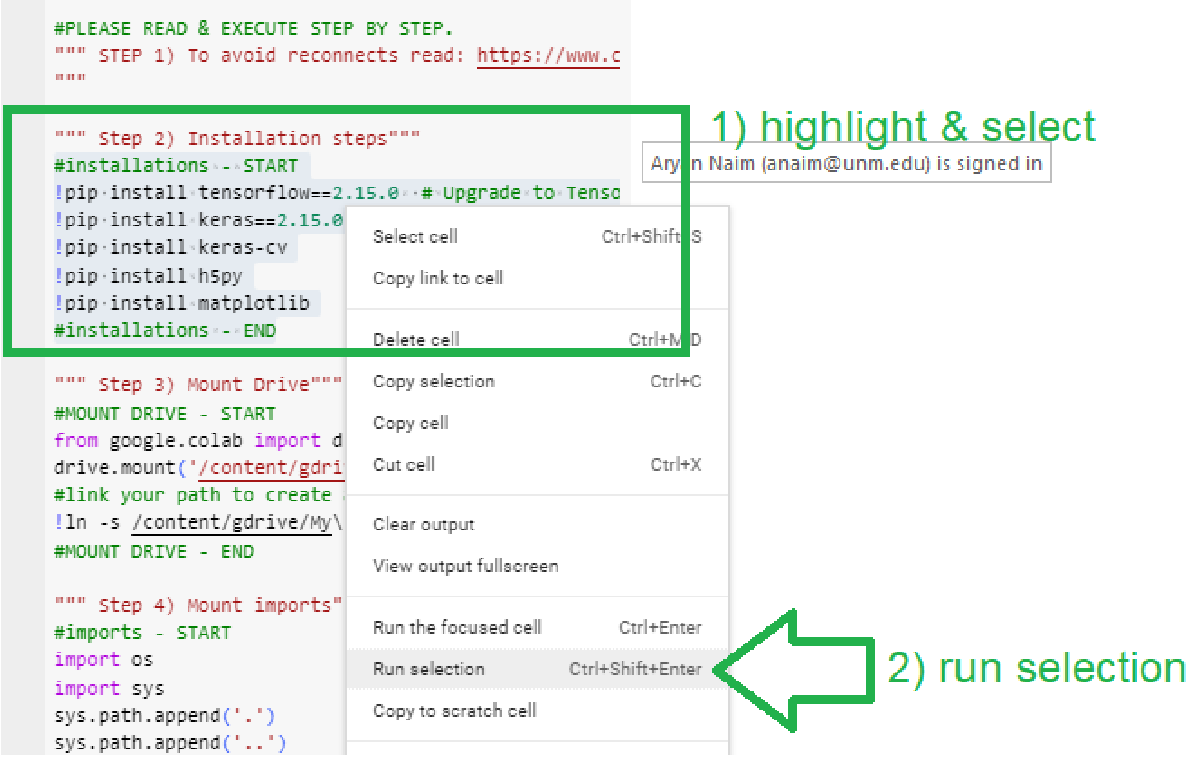
Since we are going to execute long-running tasks, we need a way to prevent Google Collab from timing out in the browser. Even if you sign up for Google Collab's "professional" version the browser will timeout after ~30 min of inactivity.

Do this hack to keep your browser alive:

```
https://www.codeease.net/programming/javascript/keep-colab-from-disconnecting
```

Step 2:

Based on the Python code in “PUBLIC_yolov8_fire_and_smoke_retrain.ipynb” install the required Python libraries.



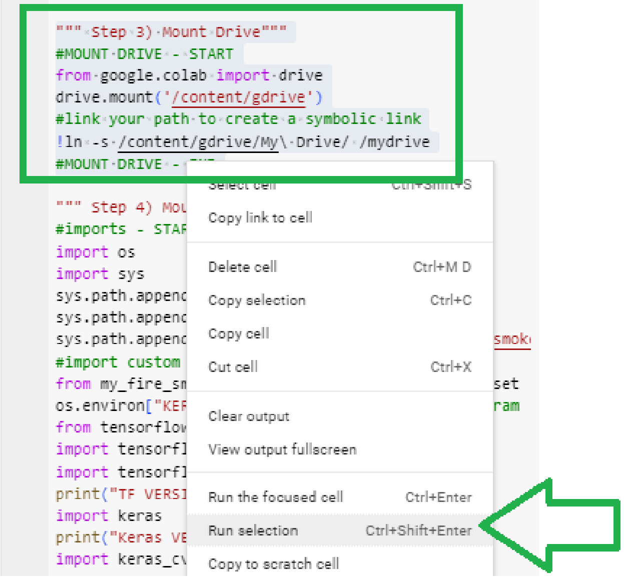
```
#PLEASE READ & EXECUTE STEP BY STEP.
""" STEP 1) To avoid reconnects read: https://www.c
"""

""" Step 2) Installation steps"""
#installations - START
!pip install tensorflow==2.15.0  # Upgrade to Tenso
!pip install keras==2.15.0
!pip install keras-cv
!pip install h5py
!pip install matplotlib
#installations - END

""" Step 3) Mount Drive"""
#MOUNT DRIVE - START
from google.colab import d
drive.mount('/content/gdri
#link your path to create
!ln -s /content/gdrive/My\
#MOUNT DRIVE - END

""" Step 4) Mount imports"
#imports - START
import os
import sys
sys.path.append('.')
sys.path.append('..')
```

1) highlight & select

2) run selection

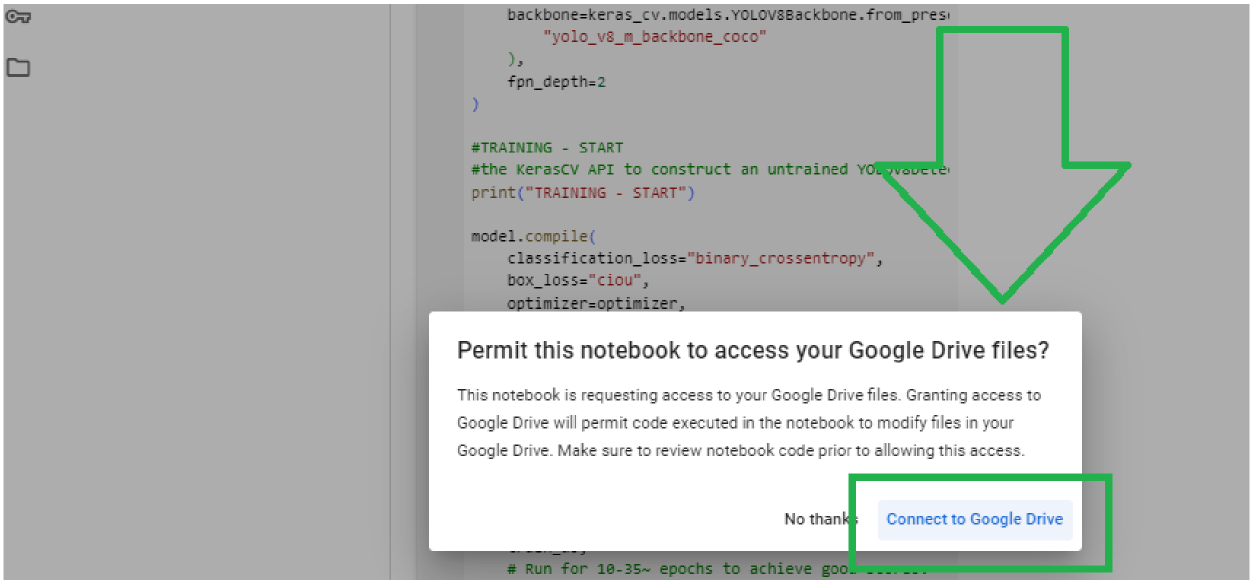
Step 3:

Mount your Google Drive



```
""" Step 3) Mount Drive"""
#MOUNT DRIVE - START
from google.colab import drive
drive.mount('/content/gdrive')
#link your path to create a symbolic link
!ln -s /content/gdrive/My\ Drive/ /mydrive
#MOUNT DRIVE -

""" Step 4) Mou
#imports - STAR
import os
import sys
sys.path.appenc
sys.path.appenc
sys.path.appenc                              smok
#import custom
from my_fire_sm                              set
os.environ["KEF                              ram
from tensorflow
import tensorfl
import tensorfl
print("TF VERSI
import keras
print("Keras VE
import keras_cv
```

You will be asked for permissions via several dialog boxes. Approve them.



```
    backbone=keras_cv.models.YOLOV8Backbone.from_pres
        "yolo_v8_m_backbone_coco"
    ),
    fpn_depth=2
)

#TRAINING - START
#the KerasCV API to construct an untrained YOLOV8Dete
print("TRAINING - START")

model.compile(
    classification_loss="binary_crossentropy",
    box_loss="ciou",
    optimizer=optimizer,
```

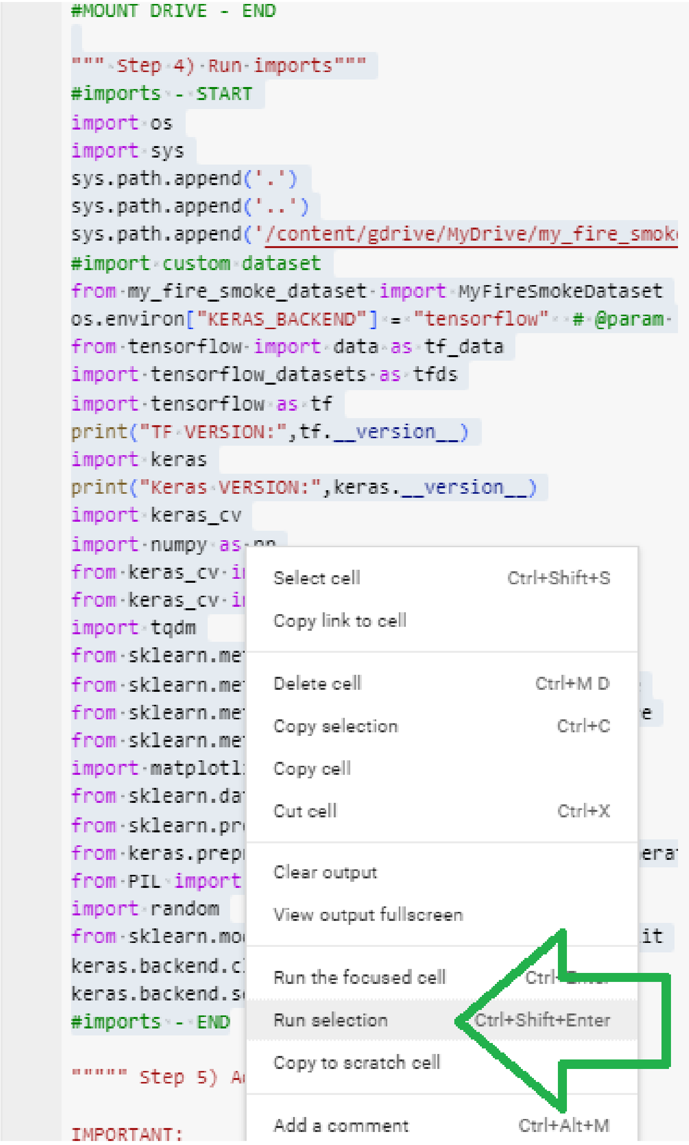
Permit this notebook to access your Google Drive files?

This notebook is requesting access to your Google Drive files. Granting access to Google Drive will permit code executed in the notebook to modify files in your Google Drive. Make sure to review notebook code prior to allowing this access.

```
    # Run for 10-35~ epochs to achieve goo
```

Continue to step 4 on the next page.

Step 4: Import the required libraries.



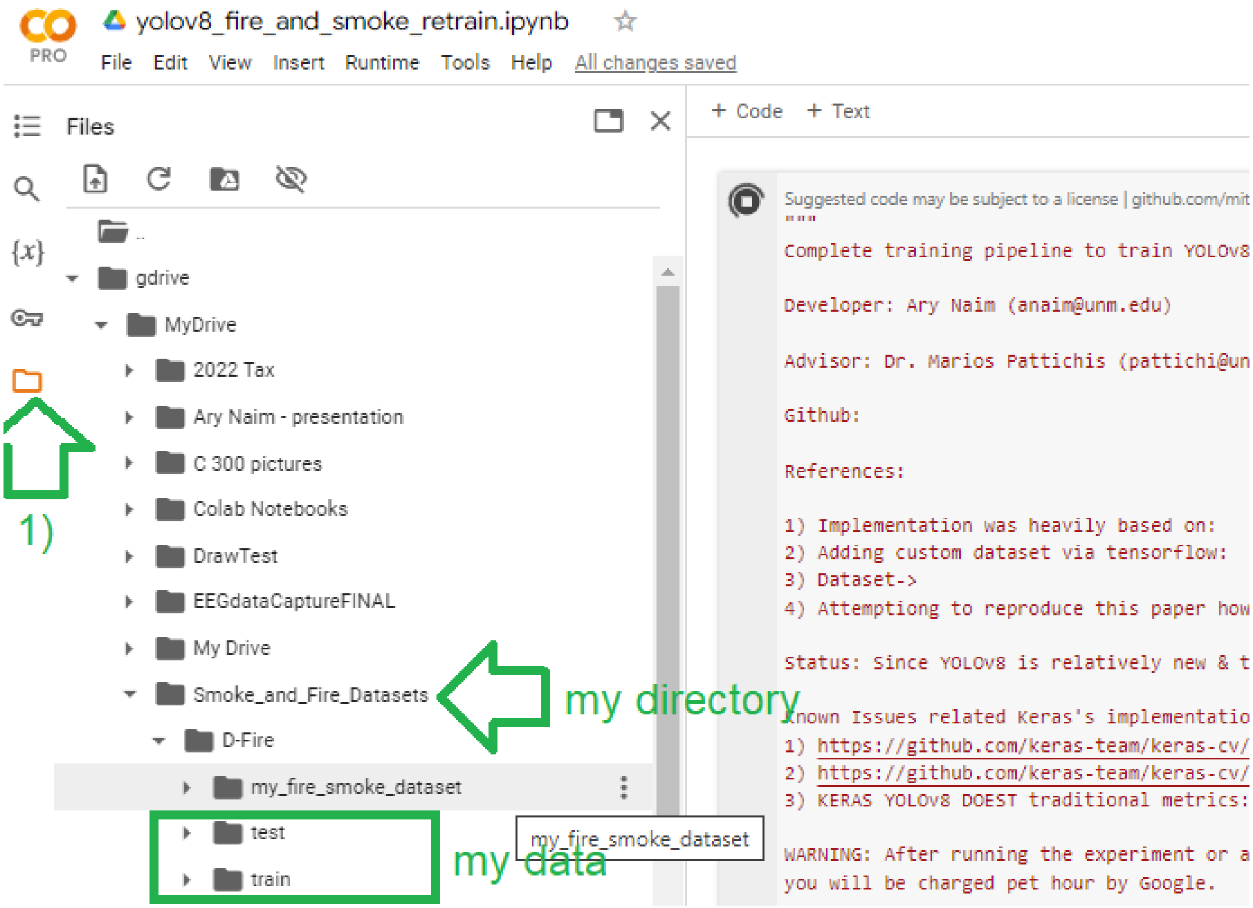
```
#MOUNT DRIVE - END

""" Step 4) Run imports"""
#imports - START
import os
import sys
sys.path.append('.')
sys.path.append('..')
sys.path.append('/content/gdrive/MyDrive/my_fire_smok
#import custom dataset
from my_fire_smoke_dataset import MyFireSmokeDataset
os.environ["KERAS_BACKEND"] = "tensorflow"  # @param
from tensorflow import data as tf_data
import tensorflow_datasets as tfds
import tensorflow as tf
print("TF VERSION:",tf.__version__)
import keras
print("Keras VERSION:",keras.__version__)
import keras_cv
import numpy as np
from keras_cv i
from keras_cv i
import tqdm
from sklearn.me
from sklearn.me
from sklearn.me
from sklearn.me
import matplotl
from sklearn.da
from sklearn.pr
from keras.prep
from PIL import
import random
from sklearn.mo
keras.backend.c
keras.backend.s
#imports - END

""""" Step 5) A

IMPORTANT:
```

There should be no errors. If there are any installation or import errors do not proceed until all the errors are resolved.

Step 5) ADVANCED - Optional for Custom Datasets.

If you are using a prepackaged dataset from Keras or just want to run the Collab notebook as is, you can skip step #5 & go to step 6.

a) Create a folder in Google Drive to upload your data.
b) Your custom data handler must implement tfds.core.DatasetBuilder. This explained the following steps.



Suggested code may be subject to a license | github.com/mit

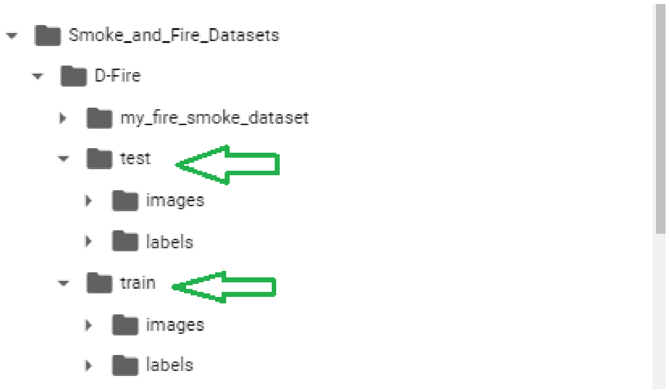
```
"""
Complete training pipeline to train YOLOv8

Developer: Ary Naim (anaim@unm.edu)

Advisor: Dr. Marios Pattichis (pattichi@un

Github:

References:

1) Implementation was heavily based on:
2) Adding custom dataset via tensorflow:
3) Dataset->
4) Attemptiong to reproduce this paper how

Status: Since YOLOv8 is relatively new & t

Known Issues related Keras's implementatio
1) https://github.com/keras-team/keras-cv/
2) https://github.com/keras-team/keras-cv/
3) KERAS YOLOv8 DOEST traditional metrics:

WARNING: After running the experiment or a
you will be charged pet hour by Google.
```

The path to my data is. You can be different.

/content/gdrive/MyDrive/Smoke_and_Fire_Datasets

Darknet/YOLOv1-4 data format:

In this specific example, the original dataset was preprocessed to be in “test” & “train” folders and each of those folders has an “image” and “labels” subfolder:

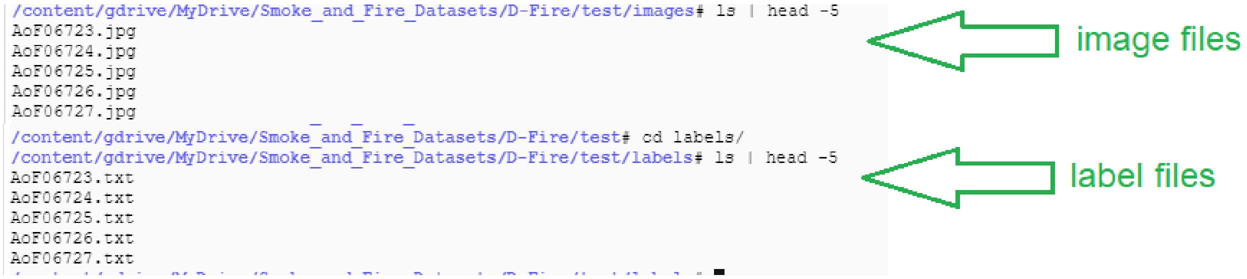


```
▾ Smoke_and_Fire_Datasets
  ▾ D-Fire
    ▸ my_fire_smoke_dataset
    ▾ test
      ▸ images
      ▸ labels
    ▾ train
      ▸ images
      ▸ labels
```

YOLO versions v1 to v4 were trained using the Darknet neural network framework (https://pjreddie.com/darknet/).

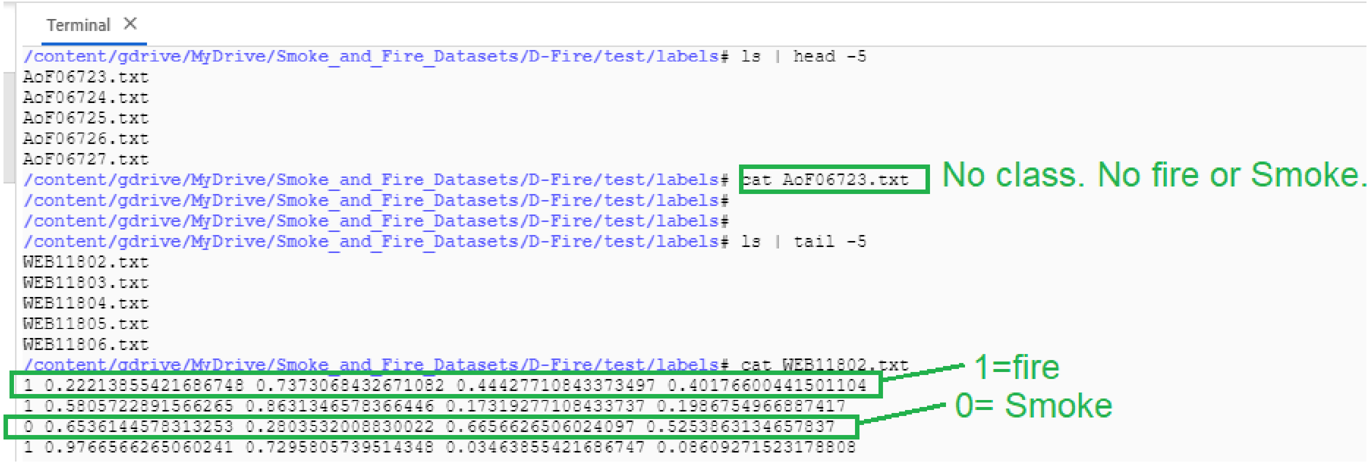
A lot of older datasets that were trained using YOLO v1-v4 have the above format of test/images, test/labels, train/images, and train/labels.

The “images” folder has image files with corresponding text label files under the “labels” folder, with the same file name.



```
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test/images# ls | head -5
AoF06723.jpg
AoF06724.jpg
AoF06725.jpg
AoF06726.jpg
AoF06727.jpg
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test# cd labels/
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test/labels# ls | head -5
AoF06723.txt
AoF06724.txt
AoF06725.txt
AoF06726.txt
AoF06727.txt
```

A closer inspection of the labels shows files that have no data indicating no classification and label files with label data in the format <class,x,y,width,height>. See below:



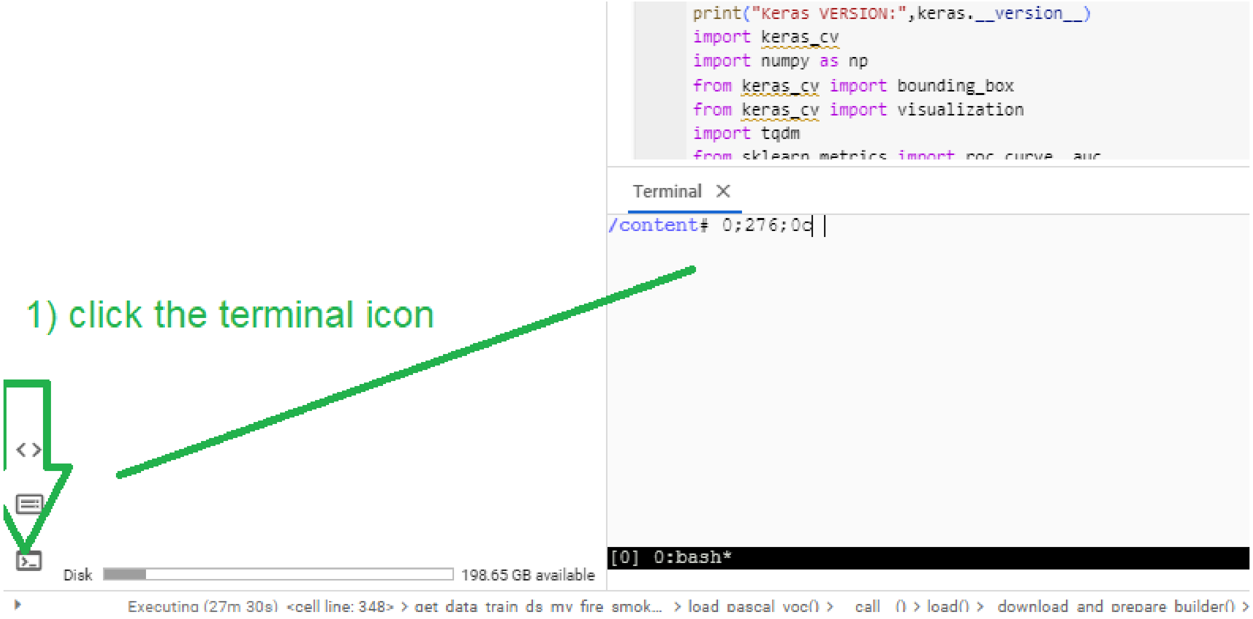
```
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test/labels# ls | head -5
AoF06723.txt
AoF06724.txt
AoF06725.txt
AoF06726.txt
AoF06727.txt
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test/labels# cat AoF06723.txt
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test/labels#
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test/labels#
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test/labels# ls | tail -5
WEB11802.txt
WEB11803.txt
WEB11804.txt
WEB11805.txt
WEB11806.txt
/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire/test/labels# cat WEB11802.txt
1 0.22213855421686748 0.7373068432671082 0.44427710843373497 0.40176600441501104
1 0.5805722891566265 0.8631346578366446 0.17319277108433737 0.1986754966887417
0 0.6536144578313253 0.2803532008830022 0.6656626506024097 0.5253863134657837
1 0.9766566265060241 0.7295805739514348 0.03463855421686747 0.08609271523178808
```

No class. No fire or Smoke.

1=fire

0= Smoke

c) Open the online browser terminal:



```
print("Keras VERSION:",keras.__version__)
import keras_cv
import numpy as np
from keras_cv import bounding_box
from keras_cv import visualization
import tqdm
```

```
/content# 0;276;0c
[0] 0:bash*
```

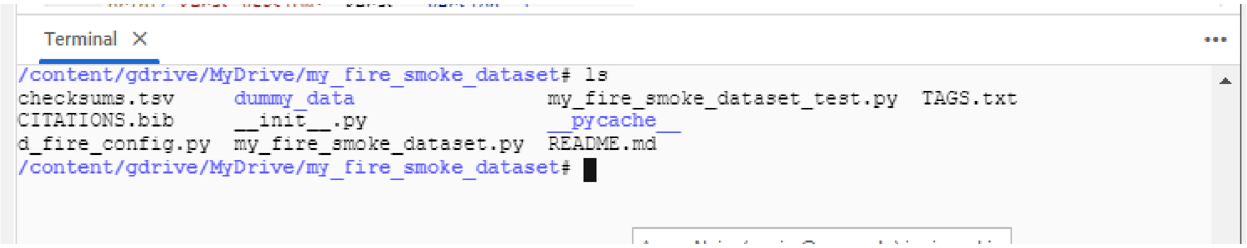
1) click the terminal icon

Disk
198.65 GB available
Executing (27m 30s) <cell line: 348> > get data train ds my fire smok... > load pascal voc() > call () > load() > download and prepare builder() >

d) Navigate to were you uploaded your data

For example:

cd /content/gdrive/MyDrive/my_fire_smoke_dataset



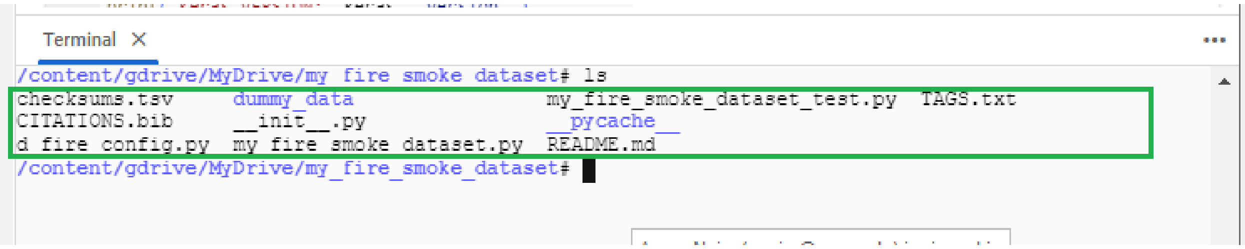
```
/content/gdrive/MyDrive/my_fire_smoke_dataset# ls
checksums.tsv     dummy_data             my_fire_smoke_dataset_test.py  TAGS.txt
CITATIONS.bib     __init__.py            __pycache__
d_fire_config.py  my_fire_smoke_dataset.py  README.md
/content/gdrive/MyDrive/my_fire_smoke_dataset#
```

All the files you see here are auto generated & you see them after executing the next command.

e) In the same path execute this command:

```
tfds new <THE_NAME_OF_YOUR_DATASET>
```

You should see bunch of files & folders appear:



```
/content/gdrive/MyDrive/my fire smoke dataset# ls
checksums.tsv     dummy_data                  my_fire_smoke_dataset_test.py  TAGS.txt
CITATIONS.bib     __init__.py                 __pycache__
d fire config.py  my fire smoke dataset.py    README.md
/content/gdrive/MyDrive/my_fire_smoke_dataset#
```

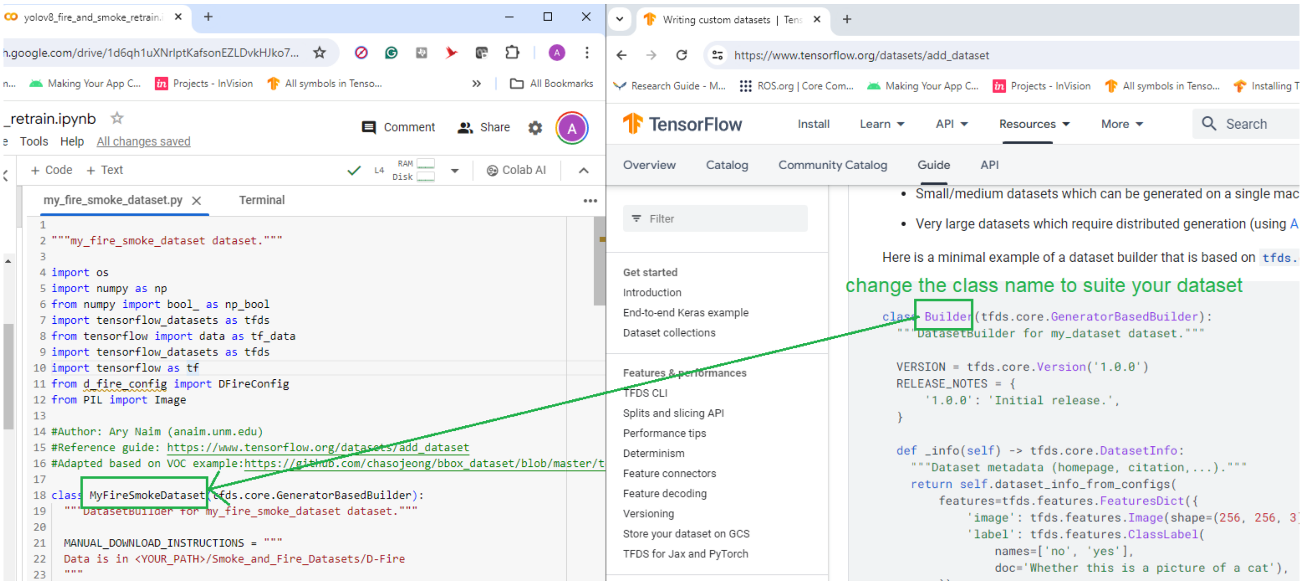
Follow this guide step by step:

READ carefully & then come back to this guide: https://www.tensorflow.org/datasets/add_dataset

f) Open the autogenerated “my_dataset_dataset_builder.py” file in one browser & keep the (https://www.tensorflow.org/datasets/add_dataset) guide open in another window as a guide.

Your custom data handling class must implement tfds.core.DatasetBuilder

Change the class name to match your dataset. Our dataset has fire & smoke images there we named it MyFireSmokeDataset:



```
"""my_fire_smoke_dataset dataset."""

import os
import numpy as np
from numpy import bool_ as np_bool
import tensorflow_datasets as tfds
from tensorflow import data as tf_data
import tensorflow_datasets as tfds
import tensorflow as tf
from d_fire_config import DFireConfig
from PIL import Image

#Author: Ary Naim (anaim.unm.edu)
#Reference guide: https://www.tensorflow.org/datasets/add_dataset
#Adapted based on VOC example:https://github.com/chasojeong/bbox_dataset/blob/master/t

class MyFireSmokeDataset(tfds.core.GeneratorBasedBuilder):
  """DatasetBuilder for my_fire_smoke_dataset dataset."""

  MANUAL_DOWNLOAD_INSTRUCTIONS = """
  Data is in <YOUR_PATH>/Smoke_and_Fire_Datasets/D-Fire
  """
```

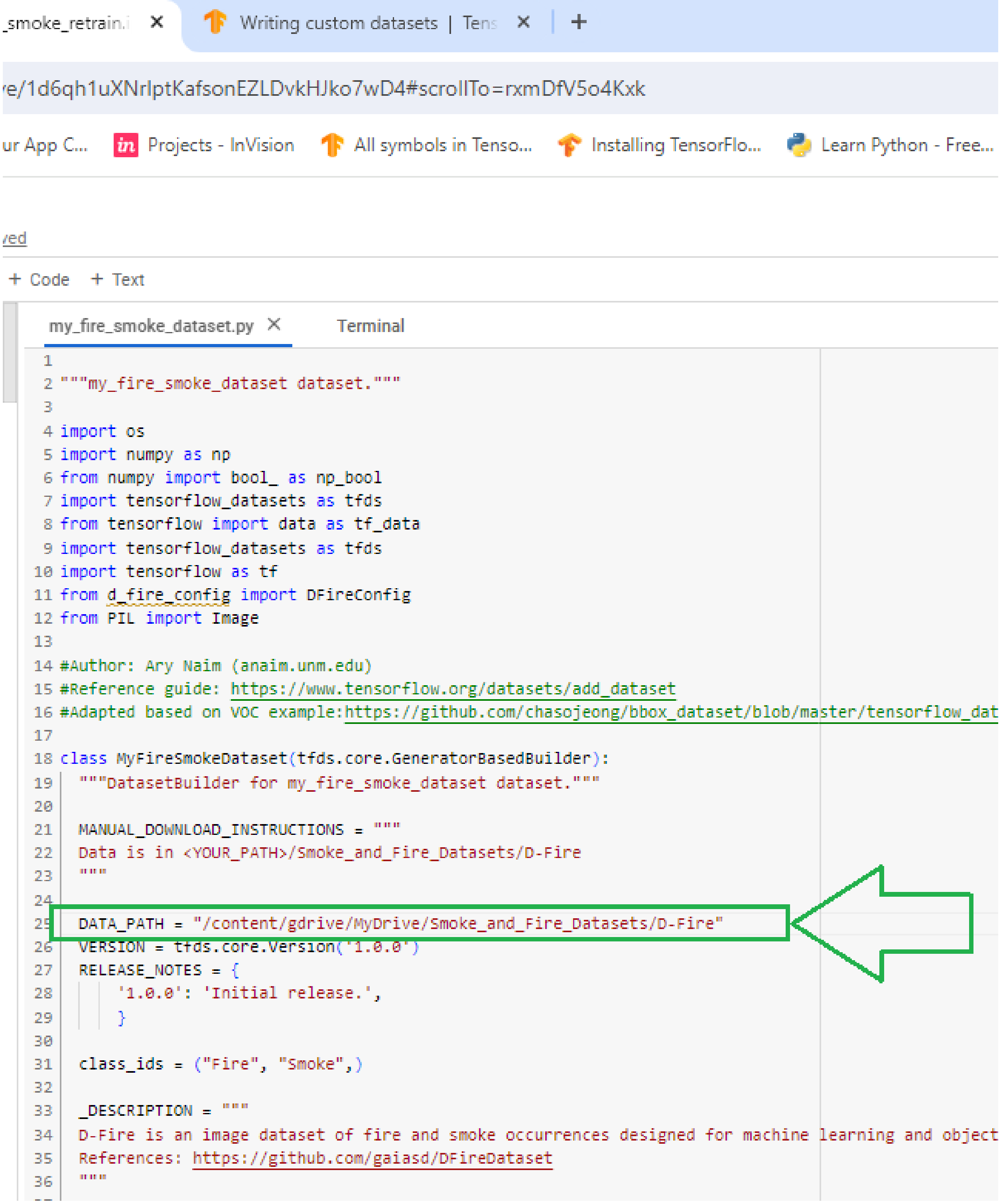
- Small/medium datasets which can be generated on a single mac
- Very large datasets which require distributed generation (using A

Here is a minimal example of a dataset builder that is based on tfds.

change the class name to suite your dataset

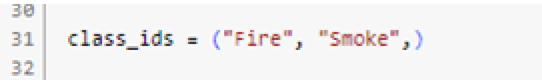
```
class Builder(tfds.core.GeneratorBasedBuilder):
  """DatasetBuilder for my_dataset dataset."""

  VERSION = tfds.core.Version('1.0.0')
  RELEASE_NOTES = {
      '1.0.0': 'Initial release.',
  }

  def _info(self) -> tfds.core.DatasetInfo:
    """Dataset metadata (homepage, citation,...)."""
    return self.dataset_info_from_configs(
        features=tfds.features.FeaturesDict({
            'image': tfds.features.Image(shape=(256, 256, 3
            'label': tfds.features.ClassLabel(
                names=['no', 'yes'],
                doc='Whether this is a picture of a cat'),
```

g) Since we are loading the data from a folder rather than from a web service, then hard code the path to your data:



```

"""my_fire_smoke_dataset dataset."""

import os
import numpy as np
from numpy import bool_ as np_bool
import tensorflow_datasets as tfds
from tensorflow import data as tf_data
import tensorflow_datasets as tfds
import tensorflow as tf
from d_fire_config import DFireConfig
from PIL import Image

#Author: Ary Naim (anaim.unm.edu)
#Reference guide: https://www.tensorflow.org/datasets/add_dataset
#Adapted based on VOC example:https://github.com/chasojeong/bbox_dataset/blob/master/tensorflow_dat

class MyFireSmokeDataset(tfds.core.GeneratorBasedBuilder):
  """DatasetBuilder for my_fire_smoke_dataset dataset."""

  MANUAL_DOWNLOAD_INSTRUCTIONS = """
  Data is in <YOUR_PATH>/Smoke_and_Fire_Datasets/D-Fire
  """

  DATA_PATH = "/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire"
  VERSION = tfds.core.Version('1.0.0')
  RELEASE_NOTES = {
      '1.0.0': 'Initial release.',
      }

  class_ids = ("Fire", "Smoke",)

  _DESCRIPTION = """
  D-Fire is an image dataset of fire and smoke occurrences designed for machine learning and object
  References: https://github.com/gaiasd/DFireDataset
  """
```

h) Change the classes to match your dataset labels:

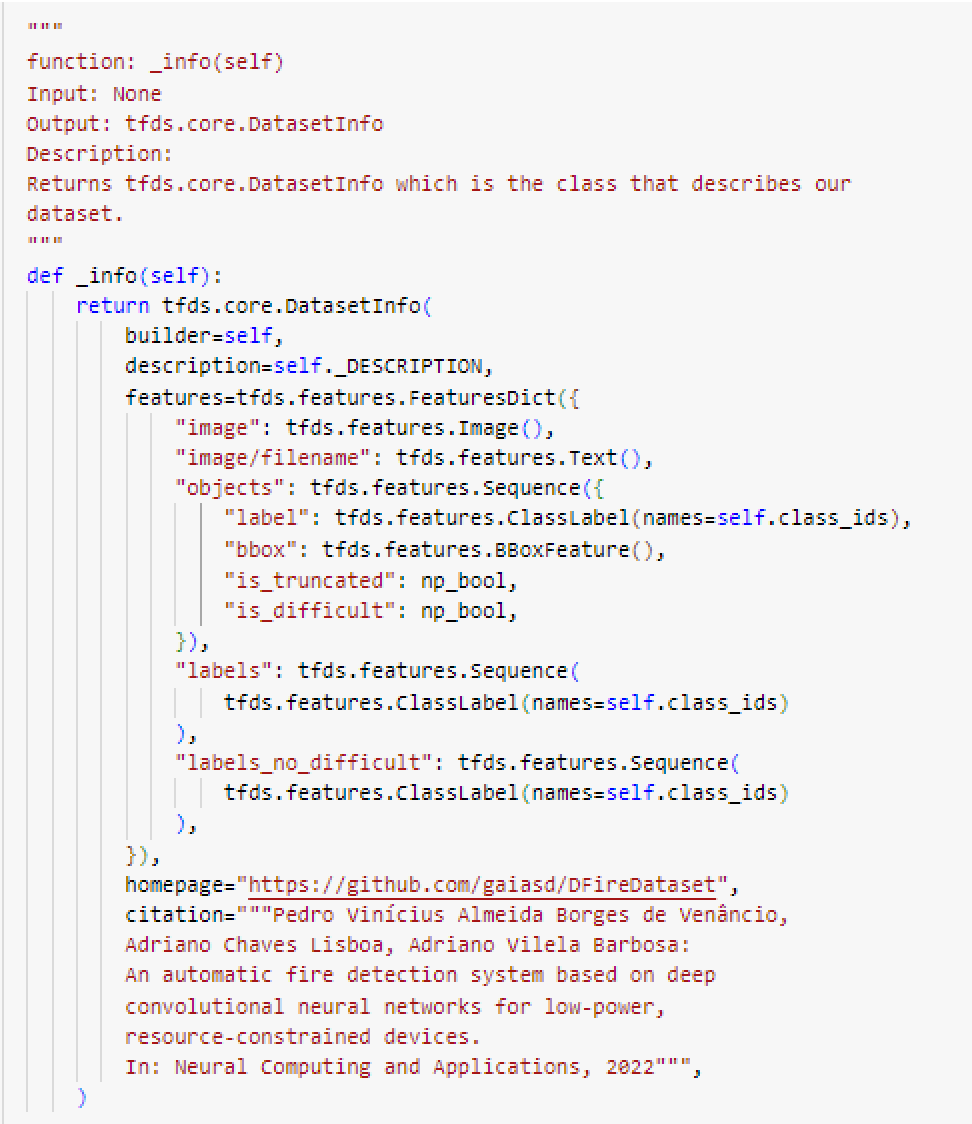


```
class_ids = ("Fire", "Smoke",)
```

i) Implement the minimum of the function required functions that are inherited from “tfds.core.GeneratorBasedBuilder”. Therefore you must implement at the minimum:

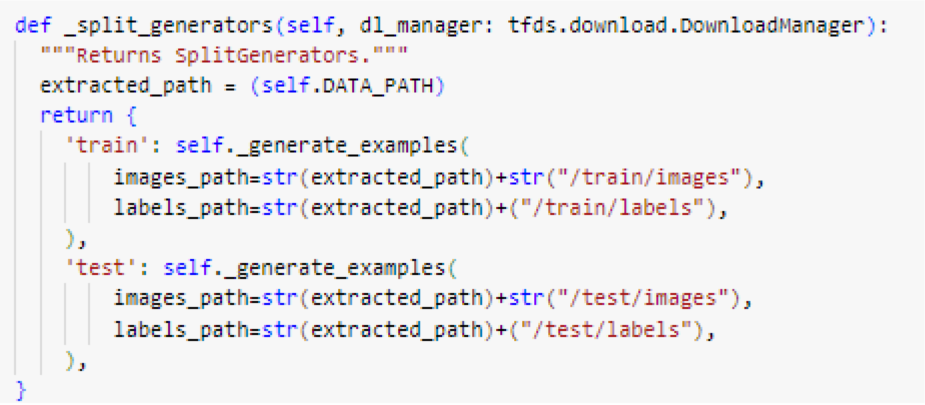
_info()
_split_generators()
_generate_examples():
_generate_example():

For example, for _info, this was our implementation:



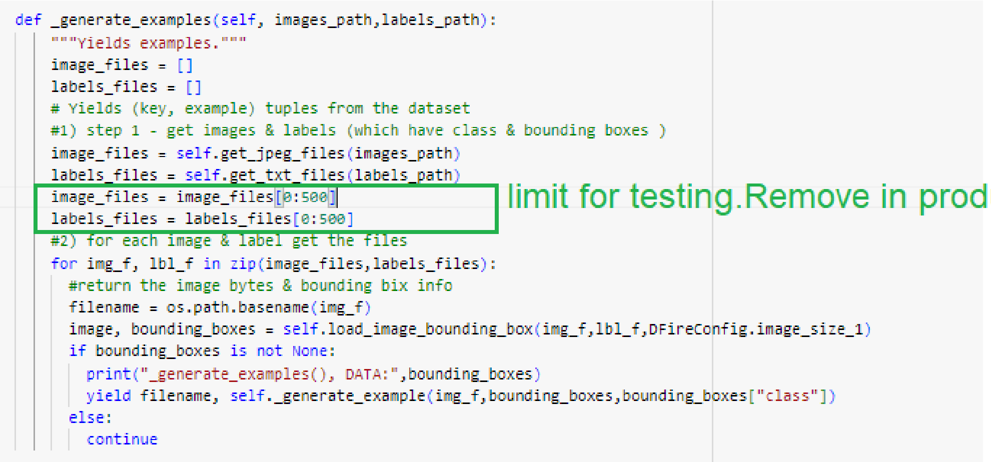
```
"""
function: _info(self)
Input: None
Output: tfds.core.DatasetInfo
Description:
Returns tfds.core.DatasetInfo which is the class that describes our
dataset.
"""
def _info(self):
    return tfds.core.DatasetInfo(
        builder=self,
        description=self._DESCRIPTION,
        features=tfds.features.FeaturesDict({
            "image": tfds.features.Image(),
            "image/filename": tfds.features.Text(),
            "objects": tfds.features.Sequence({
                "label": tfds.features.ClassLabel(names=self.class_ids),
                "bbox": tfds.features.BBoxFeature(),
                "is_truncated": np_bool,
                "is_difficult": np_bool,
            }),
            "labels": tfds.features.Sequence(
                tfds.features.ClassLabel(names=self.class_ids)
            ),
            "labels_no_difficult": tfds.features.Sequence(
                tfds.features.ClassLabel(names=self.class_ids)
            ),
        }),
        homepage="https://github.com/gaiasd/DFireDataset",
        citation="""Pedro Vinícius Almeida Borges de Venâncio,
        Adriano Chaves Lisboa, Adriano Vilela Barbosa:
        An automatic fire detection system based on deep
        convolutional neural networks for low-power,
        resource-constrained devices.
        In: Neural Computing and Applications, 2022""",
    )
```

For the function _split_generators()



```
def _split_generators(self, dl_manager: tfds.download.DownloadManager):
  """Returns SplitGenerators."""
  extracted_path = (self.DATA_PATH)
  return {
    'train': self._generate_examples(
        images_path=str(extracted_path)+str("/train/images"),
        labels_path=str(extracted_path)+("/train/labels"),
    ),
    'test': self._generate_examples(
        images_path=str(extracted_path)+str("/test/images"),
        labels_path=str(extracted_path)+("/test/labels"),
    ),
}
```

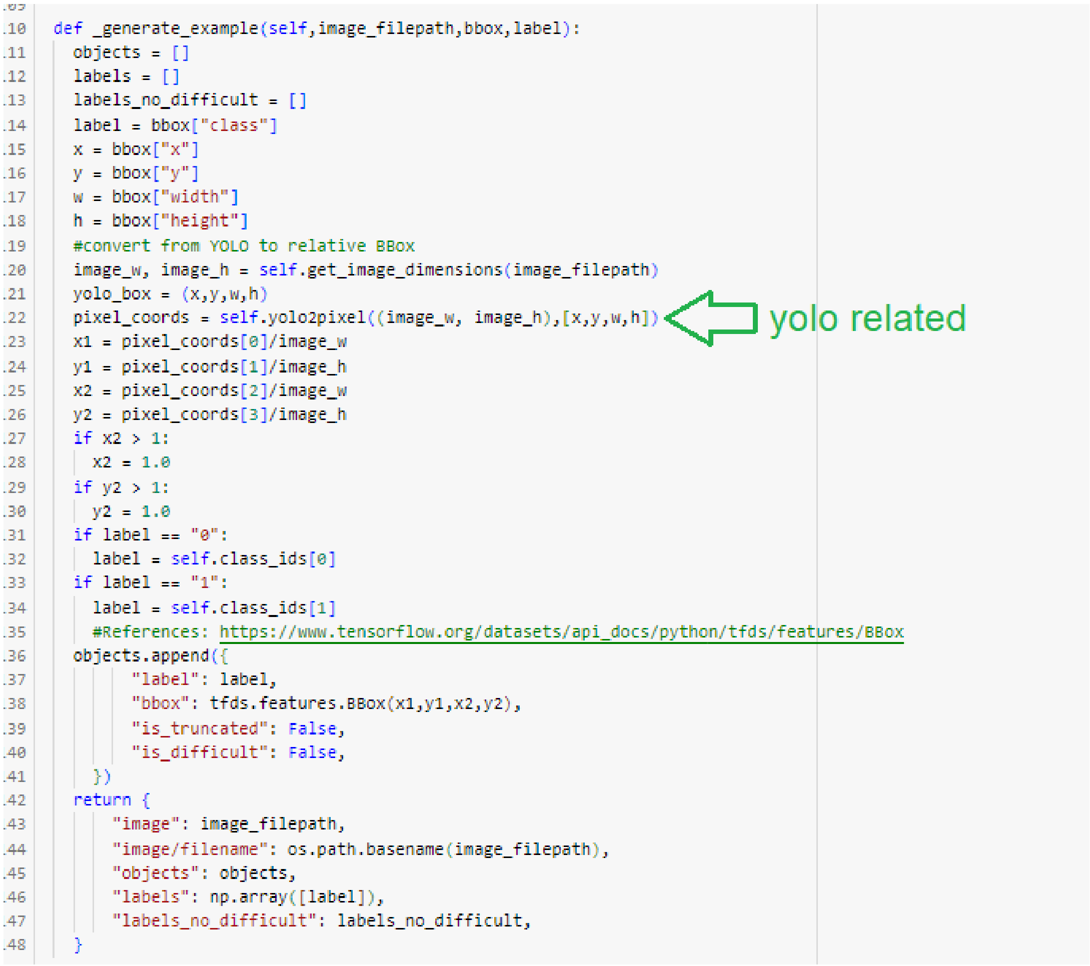
For the function _generate_examples():



```
def _generate_examples(self, images_path,labels_path):
    """Yields examples."""
    image_files = []
    labels_files = []
    # Yields (key, example) tuples from the dataset
    #1) step 1 - get images & labels (which have class & bounding boxes )
    image_files = self.get_jpeg_files(images_path)
    labels_files = self.get_txt_files(labels_path)
    image_files = image_files[0:500]
    labels_files = labels_files[0:500]
    #2) for each image & label get the files
    for img_f, lbl_f in zip(image_files,labels_files):
      #return the image bytes & bounding bix info
      filename = os.path.basename(img_f)
      image, bounding_boxes = self.load_image_bounding_box(img_f,lbl_f,DFireConfig.image_size_1)
      if bounding_boxes is not None:
        print("_generate_examples(), DATA:",bounding_boxes)
        yield filename, self._generate_example(img_f,bounding_boxes,bounding_boxes["class"])
      else:
        continue
```

limit for testing.Remove in production.

For the function _generate_example:



```
def _generate_example(self,image_filepath,bbox,label):
  objects = []
  labels = []
  labels_no_difficult = []
  label = bbox["class"]
  x = bbox["x"]
  y = bbox["y"]
  w = bbox["width"]
  h = bbox["height"]
  #convert from YOLO to relative BBox
  image_w, image_h = self.get_image_dimensions(image_filepath)
  yolo_box = (x,y,w,h)
  pixel_coords = self.yolo2pixel((image_w, image_h),[x,y,w,h])
  x1 = pixel_coords[0]/image_w
  y1 = pixel_coords[1]/image_h
  x2 = pixel_coords[2]/image_w
  y2 = pixel_coords[3]/image_h
  if x2 > 1:
    x2 = 1.0
  if y2 > 1:
    y2 = 1.0
  if label == "0":
    label = self.class_ids[0]
  if label == "1":
    label = self.class_ids[1]
    #References: https://www.tensorflow.org/datasets/api_docs/python/tfds/features/BBox
  objects.append({
      "label": label,
      "bbox": tfds.features.BBox(x1,y1,x2,y2),
      "is_truncated": False,
      "is_difficult": False,
    })
  return {
    "image": image_filepath,
    "image/filename": os.path.basename(image_filepath),
    "objects": objects,
    "labels": np.array([label]),
    "labels_no_difficult": labels_no_difficult,
  }
```

yolo related

j) After the implementation is complete then go to the terminal that’s provided in the Google Collab environment & type

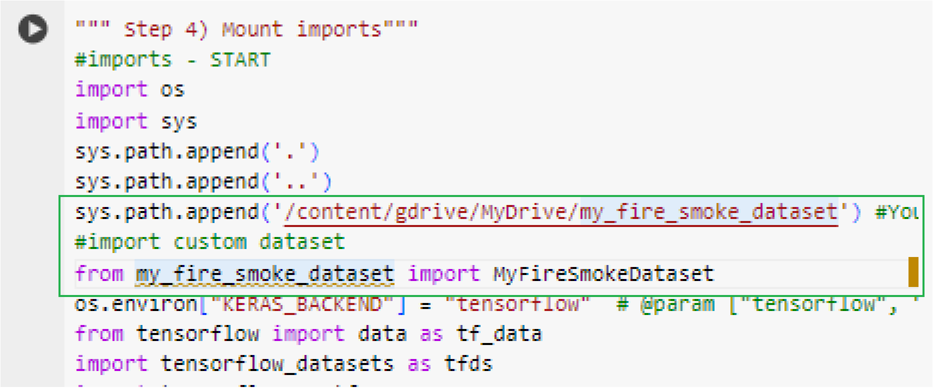
```
tfds build --register_checksums
```

This may take a while depending on the size of your dataset. It took ~10 min for a dataset of size 500 & many hours for a dataset of size 5000.

There should be no errors.

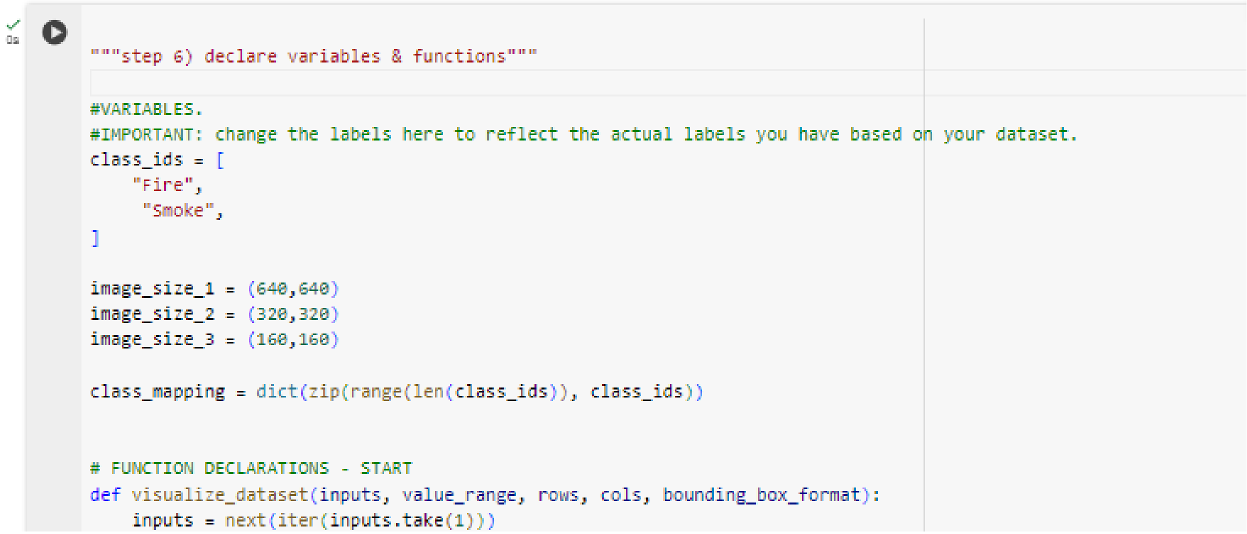
Step 6) Let's go back to the actual experiment “PUBLIC_yolov8_fire_and_smoke_retrain.ipynb”.

If you choose to do Step 5 then make sure you can import your custom Python script at the top of the imports.



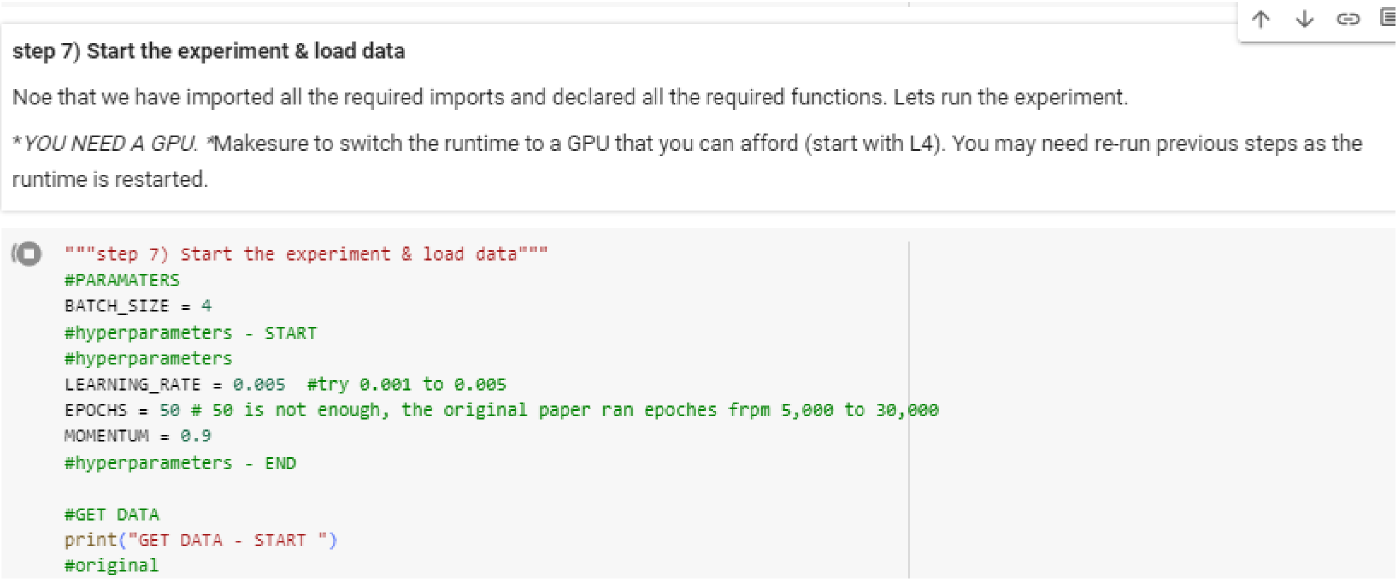
```
""" Step 4) Mount imports"""
#imports - START
import os
import sys
sys.path.append('.')
sys.path.append('..')
sys.path.append('/content/gdrive/MyDrive/my_fire_smoke_dataset') #You
#import custom dataset
from my_fire_smoke_dataset import MyFireSmokeDataset
os.environ["KERAS_BACKEND"] = "tensorflow"  # @param ["tensorflow", '
from tensorflow import data as tf_data
import tensorflow_datasets as tfds
```

Else run the section associated with declaring variables & functions.



```
"""step 6) declare variables & functions"""

#VARIABLES.
#IMPORTANT: change the labels here to reflect the actual labels you have based on your dataset.
class_ids = [
    "Fire",
     "Smoke",
]

image_size_1 = (640,640)
image_size_2 = (320,320)
image_size_3 = (160,160)

class_mapping = dict(zip(range(len(class_ids)), class_ids))


# FUNCTION DECLARATIONS - START
def visualize_dataset(inputs, value_range, rows, cols, bounding_box_format):
    inputs = next(iter(inputs.take(1)))
```

Step 7) Start the experiment & load the data:



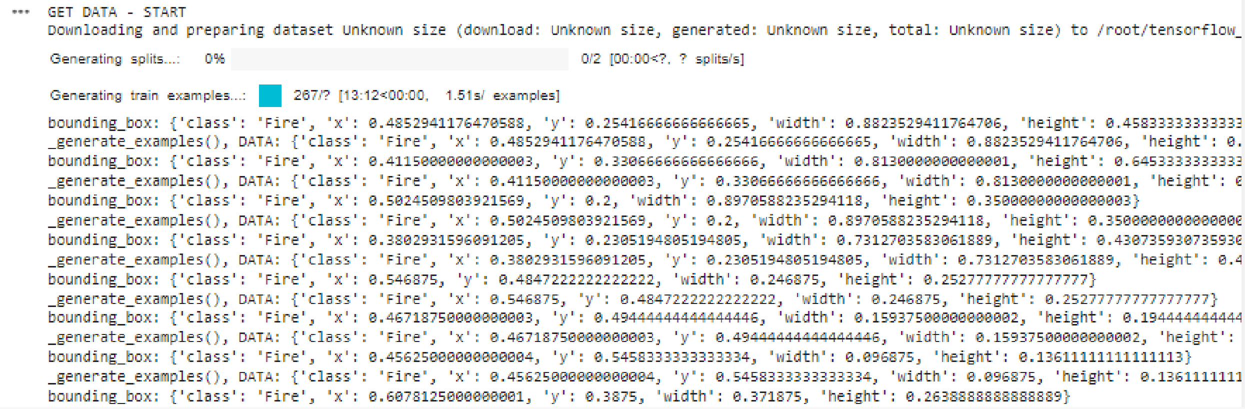
step 7) Start the experiment & load data

Noe that we have imported all the required imports and declared all the required functions. Lets run the experiment.

**YOU NEED A GPU.* *Makesure to switch the runtime to a GPU that you can afford (start with L4). You may need re-run previous steps as the runtime is restarted.

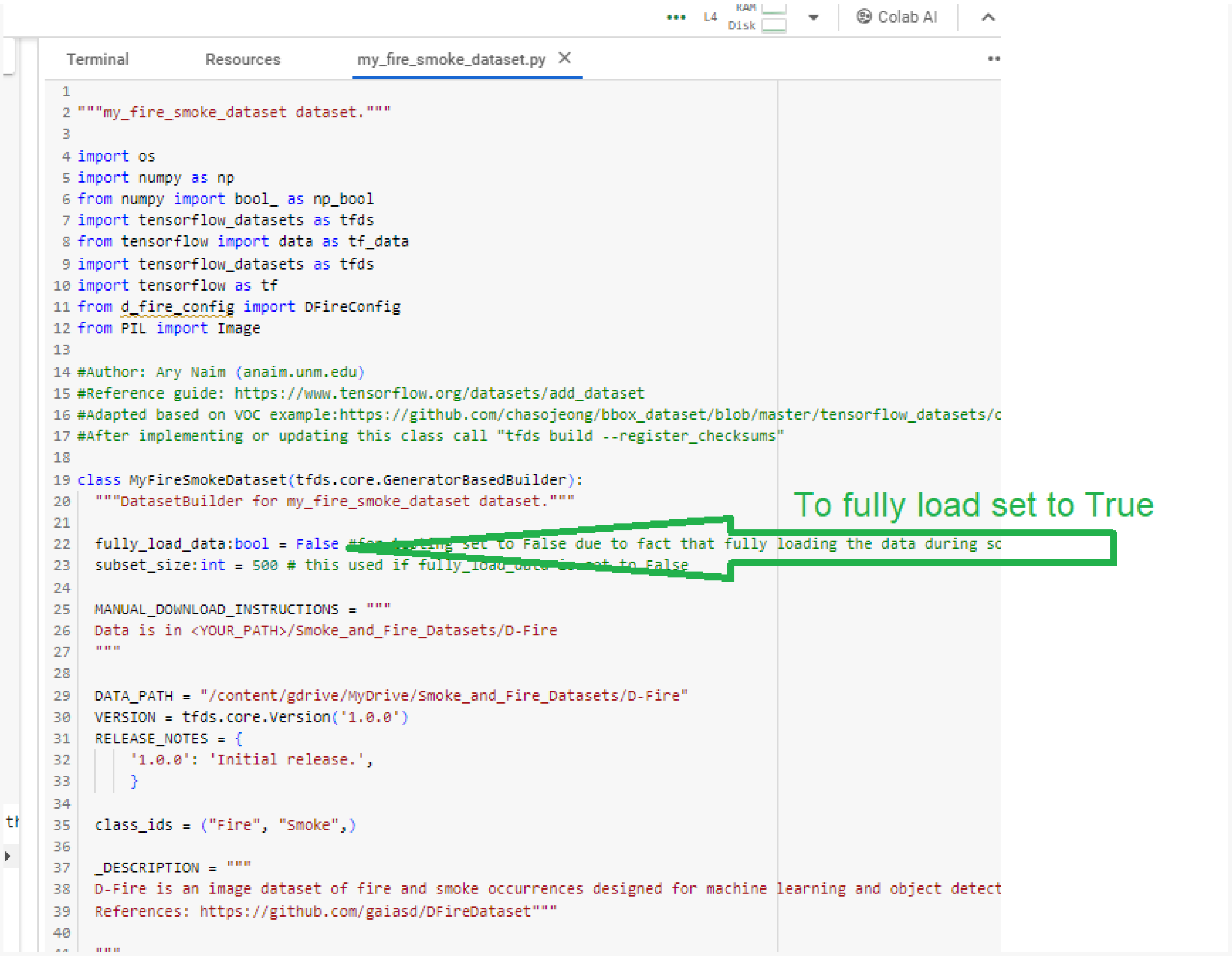
```
"""step 7) Start the experiment & load data"""
#PARAMATERS
BATCH_SIZE = 4
#hyperparameters - START
#hyperparameters
LEARNING_RATE = 0.005  #try 0.001 to 0.005
EPOCHS = 50 # 50 is not enough, the original paper ran epoches frpm 5,000 to 30,000
MOMENTUM = 0.9
#hyperparameters - END

#GET DATA
print("GET DATA - START ")
#original
```

You should start seeing the dataset being loaded. This will take a while even though we are only using 500 images which is not enough for building an accurate model.



```
GET DATA - START
Downloading and preparing dataset Unknown size (download: Unknown size, generated: Unknown size, total: Unknown size) to /root/tensorflow_
 Generating splits...:   0%                                    0/2 [00:00<?, ? splits/s]

 Generating train examples...:  ▇  267/? [13:12<00:00,  1.51s/ examples]
bounding_box: {'class': 'Fire', 'x': 0.4852941176470588, 'y': 0.25416666666666665, 'width': 0.8823529411764706, 'height': 0.45833333333333
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.4852941176470588, 'y': 0.25416666666666665, 'width': 0.8823529411764706, 'height': 0.
bounding_box: {'class': 'Fire', 'x': 0.41150000000000003, 'y': 0.33066666666666666, 'width': 0.8130000000000001, 'height': 0.64533333333333
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.41150000000000003, 'y': 0.33066666666666666, 'width': 0.8130000000000001, 'height': 0
bounding_box: {'class': 'Fire', 'x': 0.5024509803921569, 'y': 0.2, 'width': 0.8970588235294118, 'height': 0.35000000000000003}
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.5024509803921569, 'y': 0.2, 'width': 0.8970588235294118, 'height': 0.3500000000000000
bounding_box: {'class': 'Fire', 'x': 0.3802931596091205, 'y': 0.2305194805194805, 'width': 0.7312703583061889, 'height': 0.4307359307359307
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.3802931596091205, 'y': 0.2305194805194805, 'width': 0.7312703583061889, 'height': 0.4
bounding_box: {'class': 'Fire', 'x': 0.546875, 'y': 0.4847222222222222, 'width': 0.246875, 'height': 0.25277777777777777}
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.546875, 'y': 0.4847222222222222, 'width': 0.246875, 'height': 0.25277777777777777}
bounding_box: {'class': 'Fire', 'x': 0.46718750000000003, 'y': 0.49444444444444446, 'width': 0.15937500000000002, 'height': 0.194444444444
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.46718750000000003, 'y': 0.49444444444444446, 'width': 0.15937500000000002, 'height':
bounding_box: {'class': 'Fire', 'x': 0.45625000000000004, 'y': 0.5458333333333334, 'width': 0.096875, 'height': 0.13611111111111113}
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.45625000000000004, 'y': 0.5458333333333334, 'width': 0.096875, 'height': 0.1361111111
bounding_box: {'class': 'Fire', 'x': 0.6078125000000001, 'y': 0.3875, 'width': 0.371875, 'height': 0.2638888888888889}
```

If you want to load the entire 20,000 images open the Python file title “my_fire_smoke_dataset.py” and make an update in the function:

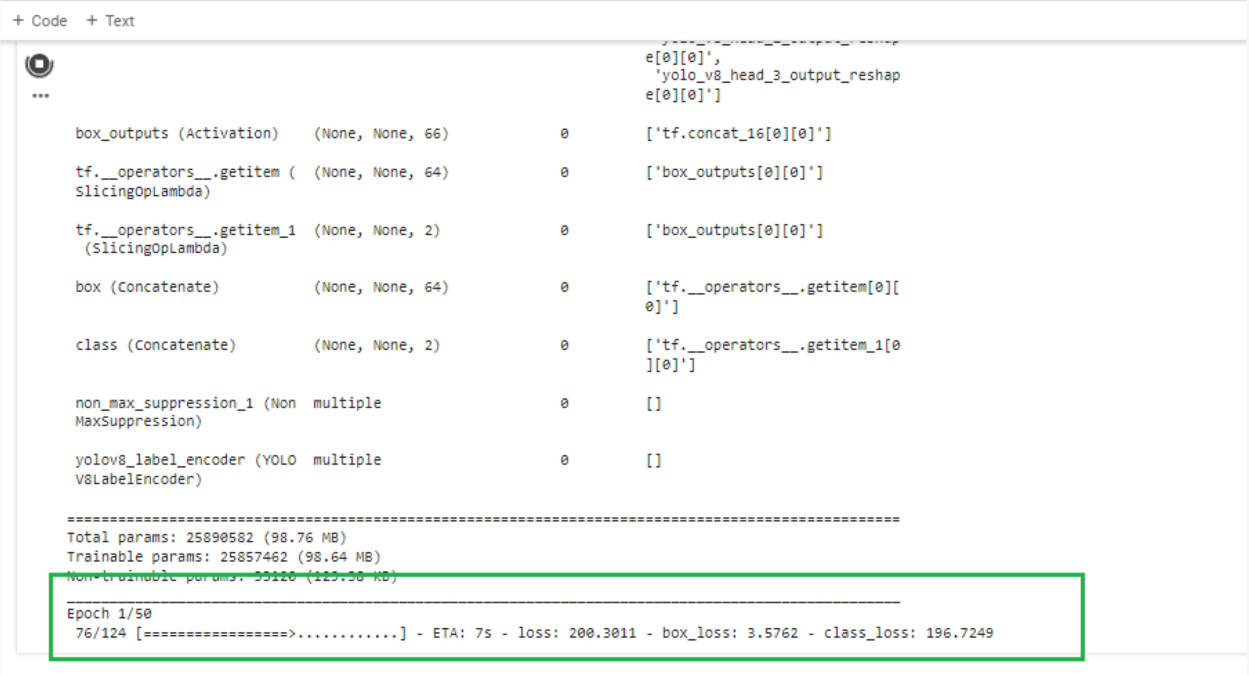


```python

"""my_fire_smoke_dataset dataset."""

import os
import numpy as np
from numpy import bool_ as np_bool
import tensorflow_datasets as tfds
from tensorflow import data as tf_data
import tensorflow_datasets as tfds
import tensorflow as tf
from d_fire_config import DFireConfig
from PIL import Image

#Author: Ary Naim (anaim.unm.edu)
#Reference guide: https://www.tensorflow.org/datasets/add_dataset
#Adapted based on VOC example:https://github.com/chasojeong/bbox_dataset/blob/master/tensorflow_datasets/c
#After implementing or updating this class call "tfds build --register_checksums"

class MyFireSmokeDataset(tfds.core.GeneratorBasedBuilder):
  """DatasetBuilder for my_fire_smoke_dataset dataset."""

  fully_load_data:bool = False #for testing set to False due to fact that fully loading the data during sc
  subset_size:int = 500 # this used if fully_load_data is set to False

  MANUAL_DOWNLOAD_INSTRUCTIONS = """
  Data is in <YOUR_PATH>/Smoke_and_Fire_Datasets/D-Fire
  """


  DATA_PATH = "/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire"
  VERSION = tfds.core.Version('1.0.0')
  RELEASE_NOTES = {
      '1.0.0': 'Initial release.',
      }

  class_ids = ("Fire", "Smoke",)

  _DESCRIPTION = """
  D-Fire is an image dataset of fire and smoke occurrences designed for machine learning and object detect
  References: https://github.com/gaiasd/DFireDataset"""

  """
```

To fully load set to True

Now you should see training results such as epoch, some metrics per each epoch, and visualizations.

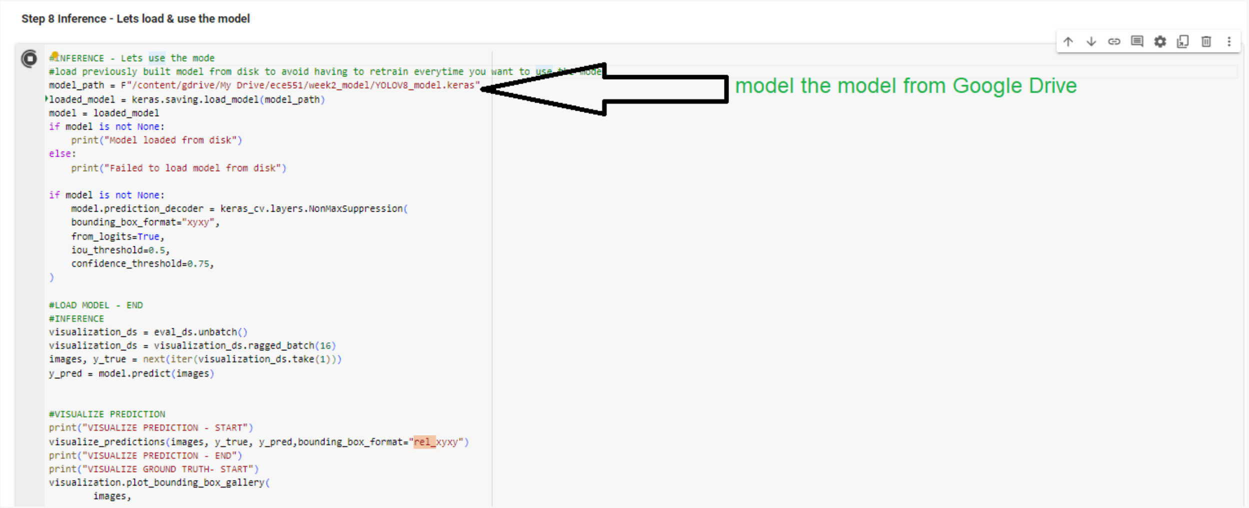
You can also see the model training per each epoch:



```
                                                                       e[0][0]',
                                                                        'yolo_v8_head_3_output_reshap
                                                                       e[0][0]']

 box_outputs (Activation)    (None, None, 66)             0         ['tf.concat_16[0][0]']

 tf.__operators__.getitem (  (None, None, 64)             0         ['box_outputs[0][0]']
 SlicingOpLambda)

 tf.__operators__.getitem_1  (None, None, 2)              0         ['box_outputs[0][0]']
  (SlicingOpLambda)

 box (Concatenate)           (None, None, 64)             0         ['tf.__operators__.getitem[0][
                                                                       0]']

 class (Concatenate)         (None, None, 2)              0         ['tf.__operators__.getitem_1[0
                                                                       ][0]']

 non_max_suppression_1 (Non  multiple                     0         []
 MaxSuppression)

 yolov8_label_encoder (YOLO  multiple                     0         []
 V8LabelEncoder)

==================================================================================================
Total params: 25890582 (98.76 MB)
Trainable params: 25857462 (98.64 MB)
Non-trainable params: 33120 (129.38 KB)
__________________________________________________________________________________________________
Epoch 1/50
 76/124 [=================>...........] - ETA: 7s - loss: 200.3011 - box_loss: 3.5762 - class_loss: 196.7249
```

Step 8) **Inference**

Execute this block to load the previously saved model and to output predictions:



Step 8 Inference - Lets load & use the model

```
#INFERENCE - Lets use the mode
#load previously built model from disk to avoid having to retrain everytime you want to use the mode
model_path = F"/content/gdrive/My Drive/ece551/week2_model/YOLOV8_model.keras"
loaded_model = keras.saving.load_model(model_path)
model = loaded_model
if model is not None:
    print("Model loaded from disk")
else:
    print("Failed to load model from disk")

if model is not None:
    model.prediction_decoder = keras_cv.layers.NonMaxSuppression(
    bounding_box_format="xyxy",
    from_logits=True,
    iou_threshold=0.5,
    confidence_threshold=0.75,
)

#LOAD MODEL - END
#INFERENCE
visualization_ds = eval_ds.unbatch()
visualization_ds = visualization_ds.ragged_batch(16)
images, y_true = next(iter(visualization_ds.take(1)))
y_pred = model.predict(images)


#VISUALIZE PREDICTION
print("VISUALIZE PREDICTION - START")
visualize_predictions(images, y_true, y_pred,bounding_box_format="rel_xyxy")
print("VISUALIZE PREDICTION - END")
print("VISUALIZE GROUND TRUTH- START")
visualization.plot_bounding_box_gallery(
        images,
```

model the model from Google Drive

You can now export your save your model to any other system that has Python, Keras, and TensorFlow installed and use that model to make predictions on input images.

DONE.