

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:

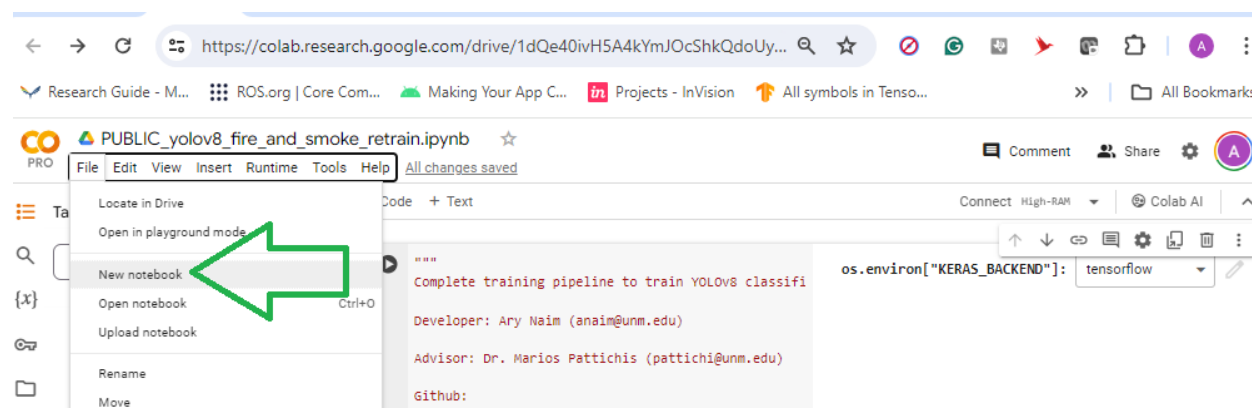
1. Create a Gmail account, in order to have a Google Drive for cloud storage.
2. Create a Google Collab account.  
This is online programming environment that gives you access to GPUs, CPUs, and can be shared with other people.

Note: Google is not free after the 1<sup>st</sup> several hours and has both monthly cost and hourly cost for GPUs.

3. Navigate to “PUBLIC\_yolov8\_fire\_and\_smoke\_retrain.ipynb” at

<https://colab.research.google.com/drive/1dQe40ivH5A4kYmJOcShkQdoUyl0sH5F0?usp=sharing>

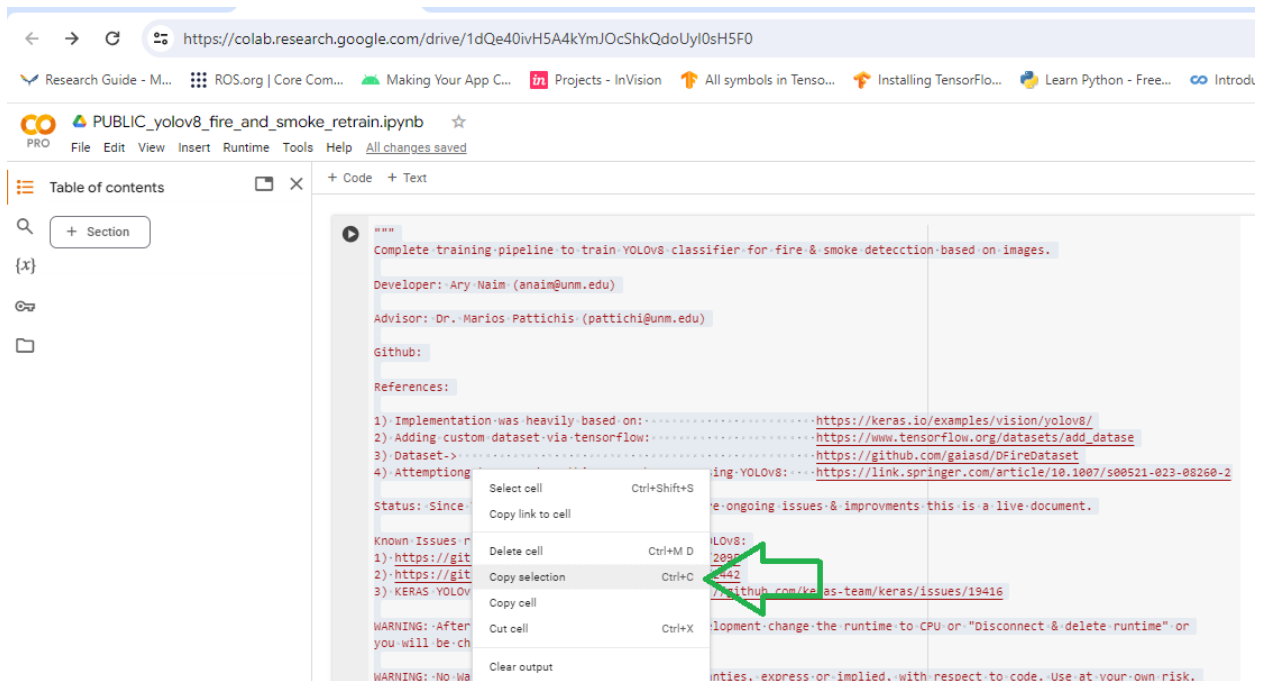
This notebook can only be viewed or read therefore in order to make modifications create your own Notebook in Collab.



**Let's start:**

### Step 0:

Copy all of the code from “PUBLIC\_yolov8\_fire\_and\_smoke\_retrain.ipynb” into your own notebook.



## Step 1:

Since we are going to run long running tasks we need a way to keep Google Collab alive even after hours of running. Even if you sign 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.

High & run the installation only:

```
#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 Tensor
!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 drive
drive.mount('/content/gdrive')
#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:

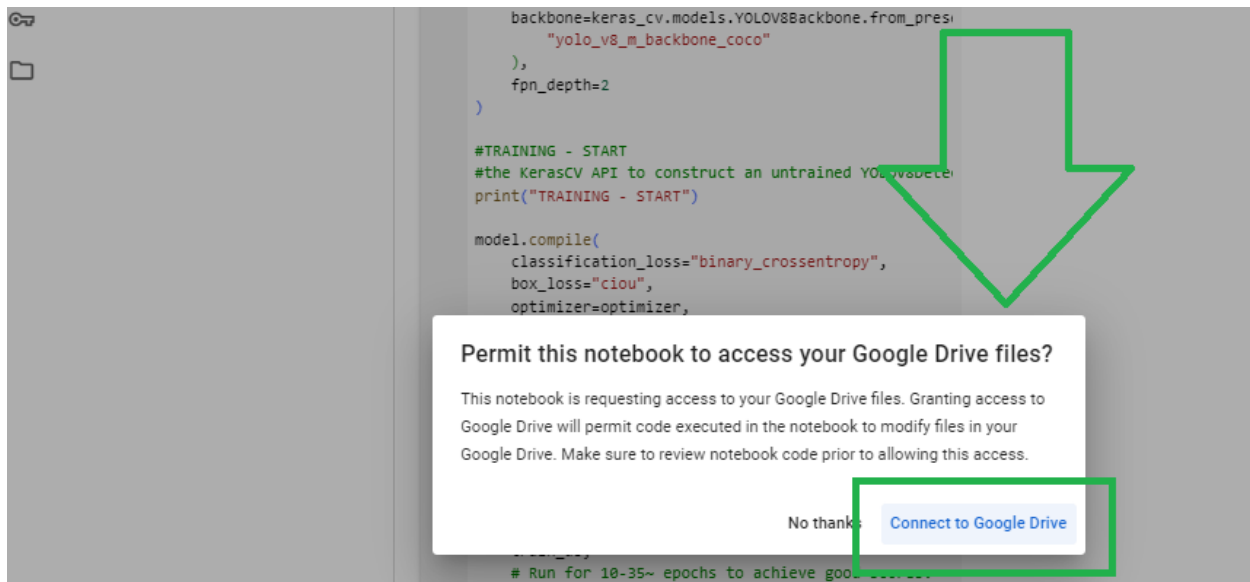
Mount your Google Drive

```
""" 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 - END

""" Step 4) Mount imports"""
#imports - START
import os
import sys
sys.path.append('.')
sys.path.append('..')
import tensorflow as tf
import keras
import keras_cv
```

2) run selection

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



Continue to step 4 on the next page.

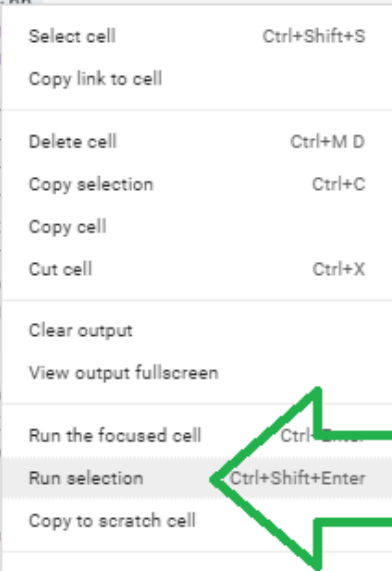
#### Step 4: Import the required libraries.

```
#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_smoke')
#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 import layers
from keras_cv import models
import tqdm
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.preprocessing import StandardScaler
from keras.preprocessing.image import ImageDataGenerator
from PIL import Image
import random
from sklearn.metrics import confusion_matrix
keras.backend.clear_session()
keras.backend.set_learning_phase(1)
#imports--END

""" Step 5) A"""

IMPORTANT:
```



Select cell	Ctrl+Shift+S
Copy link to cell	
Delete cell	Ctrl+M D
Copy selection	Ctrl+C
Copy cell	
Cut cell	Ctrl+X
Clear output	
View output fullscreen	
Run the focused cell	Ctrl+Enter
Run selection	Ctrl+Shift+Enter
Copy to scratch cell	
Add a comment	Ctrl+Alt+M

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

### Step 5) ADVANCED - Optional for Custom Datasets.

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

a) Create a folder in Google drive to upload your data & write a custom Python ()

The screenshot shows a Google Colab notebook titled "yolov8\_fire\_and\_smoke\_retrain.ipynb". The left sidebar displays the file explorer with a tree structure: "gdrive" > "MyDrive" > "Smoke\_and\_Fire\_Datasets" > "my\_fire\_smoke\_dataset". A green arrow labeled "1)" points to the "MyDrive" folder. Another green arrow labeled "my directory" points to the "Smoke\_and\_Fire\_Datasets" folder. A third green arrow labeled "my data" points to the "my\_fire\_smoke\_dataset" folder, which contains subfolders "test" and "train". The right pane shows the code editor with a suggested code block containing a license notice, a title "Complete training pipeline to train YOLOv8", developer and advisor information, a GitHub link, references, and a warning about Google charges.

```
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@unm.edu)

Github:

References:

1) Implementation was heavily based on:
2) Adding custom dataset via tensorflow:
3) Dataset->
4) Attempting 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 DOESN'T traditional metrics:

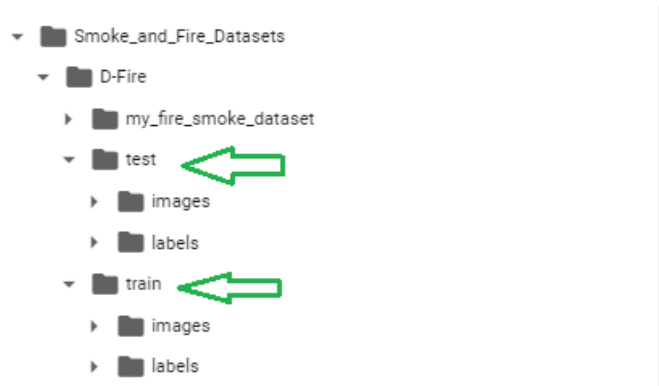
WARNING: After running the experiment or a
you will be charged pet hour by Google.
```

The path to my data is. Your 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 a “image” and “labels” subfolder:



YOLO versions v1 to v4 were trained using Darknet neural network framework (<https://pjreddie.com/darknet/>) therefore a lot of datasets have the above format of test/images, test/labels, train/images, and train/labels.

The “images” folder have images files with corresponding text label files under “labels” folder, with 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
```

image files

label files

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:

```
Terminal X
/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
No class. No fire or Smoke.

/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

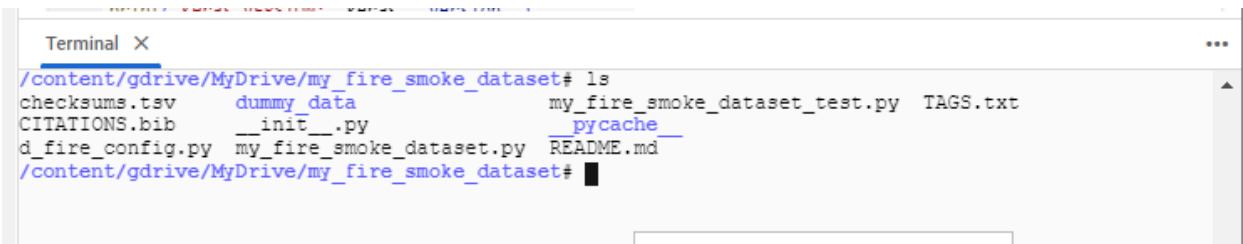
b) Open the online browser terminal:



c) Navigate to where you uploaded your data

For example:

```
cd /content/gdrive/MyDrive/my_fire_smoke_dataset
```

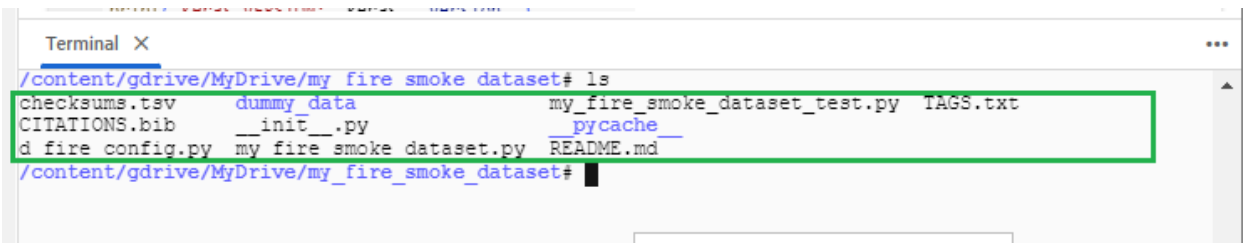


All the files you see here are auto generated & you see them after executing the next command.

d) In the same path execute this command:

```
tfds new <THE_NAME_OF_YOUR_DATASET>
```

You should see bunch of files & folders appear:





(Writing custom datasets) Follow this guide step by step:

READ carefully & then come back to this guide:

[https://www.tensorflow.org/datasets/add\\_dataset](https://www.tensorflow.org/datasets/add_dataset)

- e) Open the autogenerated “my\_dataset\_dataset\_builder.py” file in one browser & keep the ([https://www.tensorflow.org/datasets/add\\_dataset](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:

The screenshot shows a Jupyter Notebook on the left and the TensorFlow Datasets documentation on the right. The Jupyter Notebook displays the code for a custom dataset builder, `MyFireSmokeDataset`, which inherits from `tfds.core.GeneratorBasedBuilder`. A green box highlights the class name `MyFireSmokeDataset` and the base class `tfds.core.GeneratorBasedBuilder`. A green arrow points from the text "change the class name to suite your dataset" in the documentation to the highlighted class name in the code. The documentation on the right shows the `tfds.Builder` class and a minimal example of a dataset builder.

```
1 my_fire_smoke_dataset.py x Terminal
2 1
3 2 """my_fire_smoke_dataset dataset."""
4 3
5 4 import os
6 5 import numpy as np
7 6 from numpy import bool_ as np_bool
8 7 import tensorflow_datasets as tfds
9 8 from tensorflow import data as tf_data
10 9 import tensorflow_datasets as tfds
11 10 import tensorflow as tf
12 11 from d_fire_config import DFireConfig
13 12 from PIL import Image
14 13
15 14 #Author: Ary Naim (anaim.unm.edu)
16 15 #Reference guide: https://www.tensorflow.org/datasets/add_dataset
17 16 #Adapted based on VOC example:https://github.com/chasojeong/bbox_dataset/blob/master/t
18 17
19 18 class MyFireSmokeDataset(tfds.core.GeneratorBasedBuilder):
20 19     """DatasetBuilder for my_fire_smoke_dataset dataset."""
21 20
22 21     MANUAL_DOWNLOAD_INSTRUCTIONS = """
23 22     Data is in <YOUR_PATH>/Smoke_and_Fire_Datasets/D-Fire
24 23     """
```

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"),
            })
```

- f) Since we are load the data from folder rather from a web service, then hard code the path to your data:

red

+ Code + Text

my\_fire\_smoke\_dataset.py x Terminal

```
1
2 """my_fire_smoke_dataset dataset."""
3
4 import os
5 import numpy as np
6 from numpy import bool_ as np_bool
7 import tensorflow_datasets as tfds
8 from tensorflow import data as tf_data
9 import tensorflow_datasets as tfds
10 import tensorflow as tf
11 from d_fire_config import DFireConfig
12 from PIL import Image
13
14 #Author: Ary Naim (anaim.unm.edu)
15 #Reference guide: https://www.tensorflow.org/datasets/add\_dataset
16 #Adapted based on VOC example:https://github.com/chasojeong/bbox\_dataset/blob/master/tensorflow\_dat
17
18 class MyFireSmokeDataset(tfds.core.GeneratorBasedBuilder):
19     """DatasetBuilder for my_fire_smoke_dataset dataset."""
20
21     MANUAL_DOWNLOAD_INSTRUCTIONS = """
22     Data is in <YOUR_PATH>/Smoke_and_Fire_Datasets/D-Fire
23     """
24
25     DATA_PATH = "/content/gdrive/MyDrive/Smoke_and_Fire_Datasets/D-Fire"
26
27     VERSION = tfds.core.Version('1.0.0')
28     RELEASE_NOTES = {
29         '1.0.0': 'Initial release.',
30     }
31
32     class_ids = ("Fire", "Smoke",)
33
34     _DESCRIPTION = """
35     D-Fire is an image dataset of fire and smoke occurrences designed for machine learning and object
36     References: https://github.com/gaiasd/DFireDataset
37     """
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

g) Change the classes to match your datasets labels:

```
30
31 class_ids = ("Fire", "Smoke",)
32
```

- h) Implement the functions minimum 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)+str("/train/labels"),
        ),
        'test': self._generate_examples(
            images_path=str(extracted_path)+str("/test/images"),
            labels_path=str(extracted_path)+str("/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`:

```
.09
.10 def _generate_example(self, image_filepath, bbox, label):
.11     objects = []
.12     labels = []
.13     labels_no_difficult = []
.14     label = bbox["class"]
.15     x = bbox["x"]
.16     y = bbox["y"]
.17     w = bbox["width"]
.18     h = bbox["height"]
.19     #convert from YOLO to relative BBox
.20     image_w, image_h = self.get_image_dimensions(image_filepath)
.21     yolo_box = (x,y,w,h)
.22     pixel_coords = self.yolo2pixel((image_w, image_h), [x,y,w,h])
.23     x1 = pixel_coords[0]/image_w
.24     y1 = pixel_coords[1]/image_h
.25     x2 = pixel_coords[2]/image_w
.26     y2 = pixel_coords[3]/image_h
.27     if x2 > 1:
.28         x2 = 1.0
.29     if y2 > 1:
.30         y2 = 1.0
.31     if label == "0":
.32         label = self.class_ids[0]
.33     if label == "1":
.34         label = self.class_ids[1]
.35     #References: https://www.tensorflow.org/datasets/api\_docs/python/tfds/features/BBBox
.36     objects.append({
.37         "label": label,
.38         "bbox": tfds.features.BBox(x1,y1,x2,y2),
.39         "is_truncated": False,
.40         "is_difficult": False,
.41     })
.42     return {
.43         "image": image_filepath,
.44         "image/filename": os.path.basename(image_filepath),
.45         "objects": objects,
.46         "labels": np.array([label]),
.47         "labels_no_difficult": labels_no_difficult,
.48     }
```



- i) After the implementation is complete then go the terminal & 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 dataset of size 5000.

There should be no errors.

Step 6) Lets go back to the actual experiment “PUBLIC\_yolov8\_fire\_and\_smoke\_retrain.ipynb”.

If you chose to do the 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 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"""
#PARAMETERS
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...: 287/? [13:12<00:00, 1.51s/ examples]
bounding_box: {'class': 'Fire', 'x': 0.4852941176470588, 'y': 0.25416666666666665, 'width': 0.8823529411764706, 'height': 0.4583333333333333}
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.4852941176470588, 'y': 0.25416666666666665, 'width': 0.8823529411764706, 'height': 0.4583333333333333}
bounding_box: {'class': 'Fire', 'x': 0.41150000000000003, 'y': 0.33066666666666666, 'width': 0.8130000000000001, 'height': 0.6453333333333333}
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.41150000000000003, 'y': 0.33066666666666666, 'width': 0.8130000000000001, 'height': 0.6453333333333333}
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': 0.35000000000000003}
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': 0.4307359307359307}
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': 0.19444444444444444}
_generate_examples(), DATA: {'class': 'Fire', 'x': 0.46718750000000003, 'y': 0.49444444444444446, 'width': 0.15937500000000002, 'height': 0.19444444444444444}
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': 0.13611111111111113}
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:

Terminal my\_fire\_smoke\_dataset.py X

```
77     return {
78         'train': self._generate_examples(
79             images_path=str(extracted_path)+str("/train/images"),
80             labels_path=str(extracted_path)+str("/train/labels"),
81         ),
82         'test': self._generate_examples(
83             images_path=str(extracted_path)+str("/test/images"),
84             labels_path=str(extracted_path)+str("/test/labels"),
85         ),
86     }
87
88     def _generate_examples(self, images_path, labels_path):
89         """Yields examples."""
90         image_files = []
91         labels_files = []
92         # Yields (key, example) tuples from the dataset
93         #1) step 1 - get images & labels (which have class & bounding boxes )
94         image_files = self.get_jpeg_files(images_path)
95         labels_files = self.get_txt_files(labels_path)
96         image_files = image_files[0:500]
97         labels_files = labels_files[0:500]
98         #2) for each image & label get the files
99         for img_f, lbl_f in zip(image_files, labels_files):
100             #return the image bytes & bounding box info
101             filename = os.path.basename(img_f)
102             image, bounding_boxes = self.load_image_bounding_box(img_f, lbl_f, DFireConfig.image_size_1)
103             if bounding_boxes is not None:
104                 print("_generate_examples(), DATA:", bounding_boxes)
105                 yield filename, self._generate_example(img_f, bounding_boxes, bounding_boxes["class"])
106             else:
107                 continue
108
```

change or remove this.

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:

```
+ Code + Text

***
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

## Inference - Lets load & use the model

```
[13] #INFERENCE - Lets use the mode

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)
```

**DONE .**