Exploring The Wheat Dataset :

File Name :

***visualizingWheat-Images.ipynb***

CODE :

# Importing Modules :

import warnings

warnings.filterwarnings('ignore')

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import cv2 as cv2

import os

import ast

import random

import matplotlib.patches as patches

# Data – Path :

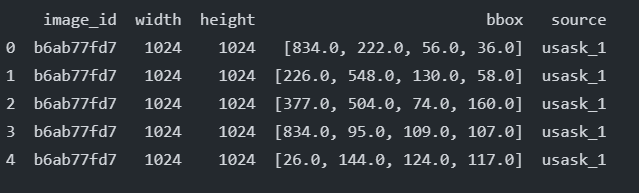
DATA\_PATH = r"E:\Final-Project\Project\global-wheat-detection"

IMG\_PATH = os.path.join(DATA\_PATH, 'train')

# Data-Frame :

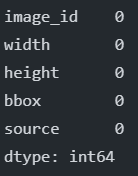
df = pd.read\_csv(os.path.join(DATA\_PATH, 'train.csv'))

print(df.head())



# Checking for NULL Values :

print(df.isnull().sum())



# Checking for Unique Values :

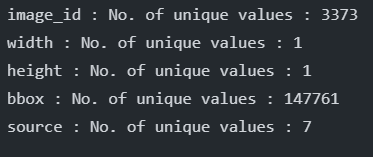
# Checking for unique values in the dataframe

def unique\_val(data):

    return f"No. of unique values : {len(data.unique())}"

for i in df.columns:

    print(f"{i} : {unique\_val(df[i])}")

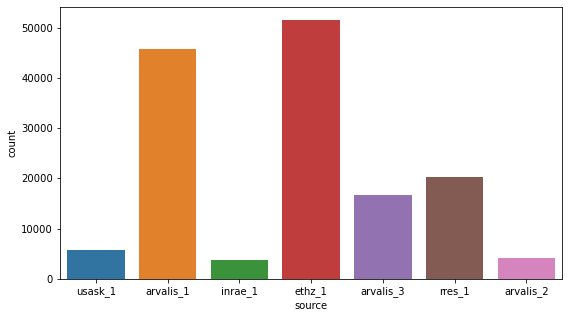


# Visualizing Source Count :

plt.figure(figsize=(9, 5))

sns.countplot(df.source)

plt.show()



# Images without Wheat heads :

unique\_image\_id = len(df['image\_id'].unique())

path = 'train'

images\_no = len(os.listdir(os.path.join(DATA\_PATH, path)))

print(f"Number of images without wheat : {images\_no - unique\_image\_id}")



# Data type of bbox :

print(type(df['bbox'][0]))

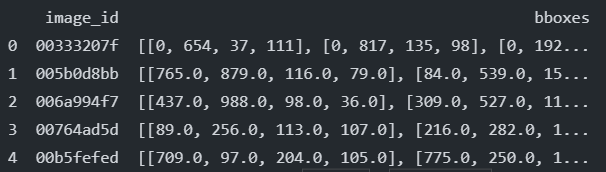


# Changing Data type :

df.bbox = df.bbox.apply(ast.literal\_eval)

df = df.groupby("image\_id")["bbox"].apply(list).reset\_index(name="bboxes")

print(df.head())



# Plotting Images :

images = os.listdir(os.path.join(DATA\_PATH, path))

def plot(path, plot\_style, num):

    img\_id = [x for x in (random.sample(range(len(images)), num))]

    for k in img\_id:

        co\_ord = df.iloc[k]['bboxes']

        img\_id = df.iloc[k]['image\_id']

        img = plt.imread(os.path.join(DATA\_PATH, path, img\_id+'.jpg'))

        figure, ax = plt.subplots(1)

        if plot\_style == 'box':

            for i in co\_ord:

                boxed\_img = img

                x, y, w, h = i

                rect = patches.Rectangle((x, y), w, h,edgecolor='r',

facecolor="none")

                ax.add\_patch(rect)

                plt.title(f"Image ID : {img\_id}")

                plt.imshow(boxed\_img)

        else:

            plt.title(f"Image ID : {img\_id}")

            ax.imshow(img)

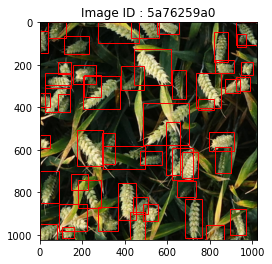
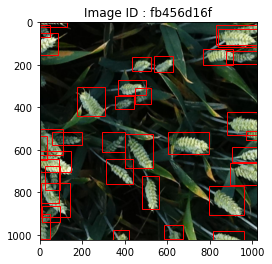
# Plotting Images without Bounding Boxes

plot(IMG\_PATH, plot\_style='normal', num=2)



# Plotting Images with Bounding Boxes

plot(IMG\_PATH, plot\_style='box',num=2)



# Downloading the YOLOV5 Model :

Command :

* git clone https://github.com/ultralytics/yolov5

# Preparing the environment :

Command :

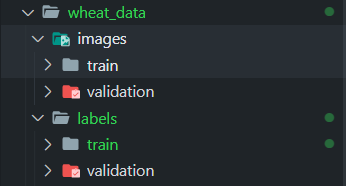
* conda activate envpytorch
* pip install -r requuirements.txt

# Pre paring Folders :

Commands :

1. cd yolov5
2. mkdir wheat\_data
3. cd wheat\_data
4. mkdir images
5. mkdir labels
6. cd images
7. mkdir train
8. mkdir validation
9. cd ../
10. cd labels
11. mkdir train
12. mkdir validation
13. cd ../

File Structure :



File Name :

***Preparing-data.py***

Code :

##########....... Preparing DATA .......##########

import pandas as pd

import numpy as np

import os

import ast

from tqdm import tqdm

import shutil

from sklearn import model\_selection

import sys

DATA\_PATH = r"E:\Final-Project\Project\global-wheat-detection"

ROOT\_DIR = r"E:\Final-Project\Project\yolov5\wheat\_data"

OUTPUT\_PATH = ROOT\_DIR

def process\_data(data, data\_type="train"):

  for \_, row in tqdm(data.iterrows(), total=len(data)):

    image\_name = row['image\_id']

    bounding\_boxes = row['bboxes']

    yolo\_data = []

    for bbox in bounding\_boxes:

      x = bbox[0]

      y = bbox[1]

      w = bbox[2]

      h = bbox[3]

      x\_center = x + w / 2

      y\_center = y + h / 2

      x\_center /= 1024.0

      y\_center /= 1024.0

      w /= 1024.0

      h /= 1024.0

      yolo\_data.append([0, x\_center, y\_center, w, h])

    yolo\_data = np.array(yolo\_data)

    np.savetxt(

        os.path.join(OUTPUT\_PATH, f"labels\\{data\_type}\\{image\_name}.txt"),

        yolo\_data,

        fmt=["%d", "%f", "%f", "%f", "%f"]

    )

    shutil.copyfile(

        os.path.join(DATA\_PATH, f"train\\{image\_name}.jpg"),

        os.path.join(OUTPUT\_PATH, f"images\\{data\_type}\\{image\_name}.jpg")

    )

if \_\_name\_\_ == "\_\_main\_\_":

  df = pd.read\_csv(os.path.join(DATA\_PATH, "train.csv"))

  df.bbox = df.bbox.apply(ast.literal\_eval)

  df = df.groupby("image\_id")["bbox"].apply(list).reset\_index(name="bboxes")

  df\_train, df\_valid = model\_selection.train\_test\_split(

      df,

      test\_size=0.1,

      random\_state=42,

      shuffle=True

  )

  df\_train = df\_train.reset\_index(drop=True)

  df\_valid = df\_valid.reset\_index(drop=True)

  process\_data(df\_train, data\_type="train")

  process\_data(df\_valid, data\_type="validation")

# Making data file:

Create : wheat.yaml

File Contents :

train: wheat\_data/images/train #Training Folder Path

val: wheat\_data/images/validation #Validation Folder Path

nc: 1 # Number of Classes

names: ["wheat"] #Name of class

# Running the model on the command prompt within the intended environment

Command :

* conda activate envpytorch
* python train.py --img 1024 --batch 2 --epochs 15 --data wheat.yaml --cfg models/yolov5s.yaml --name wheat

If everything is followed correctly Model should Run Properly.

Further we can visualize the outcomes using tensorboard:

Command :

tensorboard --logdir runs/