The train.csv file has the following columns:

|  |  |
| --- | --- |
| Column Name | Meaning |
| class | Class code of the pollution type |
| image\_path | The name of the .jpg image |
| name | Name of visual pollution type |
| xmax | Xmax coordinate of bounding box |
| xmin | Xmin coordinate of bounding box |
| ymax | Ymax coordinate of bounding box |
| ymin | Ymin coordinate of bounding box |

Task: Build an ML model that can detect various objects in images.

The dataset folder has the following files:

* Images: 9966 image files
* train.csv: 19950 \* 7
* text.csv: 2092 \* 1
* sample\_submission.csv

Evaluation metric: score = 100 \* mean\_average\_precision

Result submission guidelines:

* The submission file must be submitted in .csv format.
* The submission file must contain predictions for all 2092 images in the train.csv file.
* Have correct image paths per the test file.
* Have the names of columns as provided in the sample\_submission.csv file.

It looks like the final submission only needs to be a csv file, so my code can be whatever I need it to be.

Visual pollution types, class code, and # images in training per type:

|  |  |  |
| --- | --- | --- |
| Class | Name | # Images in train.csv |
| 0 | GRAFFITI | 1124 |
| 1 | FADED\_SIGNAGE | 107 |
| 2 | POTHOLES | 2625 |
| 3 | GARBAGE | 8597 |
| 4 | CONSTRUCTION\_ROAD | 2730 |
| 5 | BROKEN\_SIGNAGE | 83 |
| 6 | BAD\_STREETLIGHT | 1 |
| 7 | BAD\_BILLBOARD | 1555 |
| 8 | SAND\_ON\_ROAD | 748 |
| 9 | CLUTTER\_SIDEWALK | 2253 |
| 10 | UNKEPT\_FACADE | 127 |

Here are the contents of each file:

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application, table, Excel

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

It looks like each image is used more than once, so I think I will have to break down the image into different parts.

I have a few steps:  
1. Get the individual images (break down the image using the coordinates) per pollution type.

2. Use **Teachable Machine** to load all these images into separate classes

3. Using the test image, get the classification

I did all of the 3 above, but it looks like I might have to break the images down to the provided xy coordinates.

I will do this tomorrow. In the meantime, I will ask a question about whether we need to not only classify the type of image but also specify the xy coordinates of the test images to input into the sample\_submission.csv file.

In the meantime, it looks like the bounds of the xy dimensions are not accurate, here is the response given by the Admin on the Discussion board:

“Hello participants! We are incredibly excited about getting so many teams working on Theme 1 and look forward to your results! Let's been a bit of confusion regarding the standard we used to mark the data and realize how this can be confusing.

The solution to this is very simple, take the xmax, ximin, ymax and ymin values and multiply them by 2. Once you do that, you should see the bounding boxes adjusted against the high-resolution imagery you have. Worried about values in xmin or ymin that have negative values? No need to worry. It's common that the bounding boxes go above the image pixels, in particular with objects that cover more than what is visible. You can manage this however you see fit, up to you!

Please take note: When delivering a results.csv file, assume that the image width is 1920/2 (960 pixels wide).

Here is some sample code attached that might help you verify the Bounding Boxes.

import cv2

def plot\_one\_box(x, img, color=None, label=None, line\_thickness=None, Inverted=False):

# Plots one bounding box on image img

tl = line\_thickness or 2 # line thickness

c1, c2 = (int(x[0]), int(x[1])), (int(x[2]), int(x[3]))

cv2.rectangle(img, c1, c2, color, thickness=tl)

if label:

tf = tl # font thickness

t\_size = cv2.getTextSize(label, 0, fontScale=tl / 2, thickness=tf)[0]

if Inverted == True:

c1 = c2

c2 = c1[0] + t\_size[0], c1[1] - t\_size[1] - 3

else:

c2 = c1[0] + t\_size[0], c1[1] - t\_size[1] - 3

cv2.rectangle(img, c1, c2, color, -1) # filled

cv2.putText(

img,

label,

(c1[0], c1[1] - 2),

0,

tl / 2,

[225, 255, 255],

thickness=tf,

lineType=cv2.LINE\_AA,

)

# Using readlines()

file1 = open('train.csv', 'r')

Lines = file1.readlines()

count = 0

# Strips the newline character

for line in Lines:

if count == 0:

count += 1

continue

file\_id\_path = line.split(',')[2]

# open image in cv2

img = cv2.imread("images/" + file\_id\_path)

h, w, c = img.shape

cat = line.split(',')[3]

xmax = int(line.split(',')[4]) \* 2

xmin = int(line.split(',')[5]) \* 2

ymax = int(line.split(',')[6]) \* 2

ymin = int(line.split(',')[7]) \* 2

# plot the box

plot\_one\_box([xmin, ymin, xmax, ymax], img, color=(0, 255, 0), label=cat, line\_thickness=2)

# save the image

# you might need to create the folder "drawn" first!

cv2.imwrite("drawn/" + file\_id\_path, img)

print("Line {}: {}".format(count, line.strip()))

count += 1

“

My question is below:

“Hello, I have a question regarding the submission csv format for Theme #1.

The sample\_submission.csv file has similar columns to what the train.csv file has. Can we change the format of the submission csv file to have different columns so that it includes more information from the ML model? Thanks.”