Analyze Semantic Segmentation dataset

The dataset that available for training is consisting of 2621 images with masks included. We Splitted the dataset 79%, 21% for training, And validation set respectevly.

In this notebook, I am gonna find answers for the following questions:

- Q1: does the image sizes uniformly distributed? how are the variations in images resolution will effect on the development process?
- Q2: what are the most dominant classes?
- Q3: what are the average number of pixels of each class?
- Q4: where each class mostly located?

Step 1: Explore the dataset visually

since the dataset is kindly small, we could inspect it!

```
%matplotlib inline

from easyimages import EasyImageList
from torch.utils.data import DataLoader,Dataset
from dataclasses import dataclass
import pandas as pd
import numpy as np
from os.path import join
import cv2
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats as st
import torch
import albumentations as A
from rgb_segmentation import get_corresponding_Color
import dataframe_image as dfi
from matplotlib.colors import LinearSegmentedColormap

image_list = EasyImageList.from_folder(r"C:\Users\falmasridev\
Documents\opencv_courses\c2\opencv-pytorch-segmentation-project-
round2\imgs\imgs")

image_list.html(sample=100,size=50)
```

```
<IPython.core.display.HTML object>
```

prepare data

```
#we gonna use the dataset class from torch for loading images with
ease.
```

```
class SemSegDataset(Dataset):

    def
__init__(self,data_path,images_folder,masks_folder,csv_path,dataset_ty
pe,num_classes,validset_ratio,transform=None,class_names=None):
        self.data_path = data_path
        self.images_folder = images_folder
        self.masks_folder = masks_folder
        self.csv_path = csv_path
        self.dataset_type = dataset_type
        self.num_classes = num_classes
        self.transform = transform
        self.class_names = class_names


        self.image_ids =
pd.read_csv(join(data_path,csv_path)).astype('str')
        if dataset_type == 'train' or dataset_type == 'valid':

            train_set = self.image_ids.sample(frac=1-validset_ratio)
            valid_set = self.image_ids.drop(train_set.index)



            if dataset_type == 'train':
                self.dataset = train_set
            else:
                self.dataset = valid_set

        elif dataset_type == 'test':
            self.dataset = self.image_ids
        else:
            raise Exception("Wrong dataset type")

        self.dataset.reset_index(inplace=True,drop=True)


    def __len__(self):
        return len(self.dataset)

    def __getitem__(self,index):
        img =
cv2.cvtColor(cv2.imread(f"{join(self.images_folder,self.dataset.iloc[i
ndex]['ImageID'])}.jpg",cv2.IMREAD_UNCHANGED),cv2.COLOR_BGR2RGB)

        if self.dataset_type != 'test':
            mask =
cv2.imread(f"{join(self.masks_folder,self.dataset.iloc[index]
['ImageID'])}.png",cv2.IMREAD_UNCHANGED)
```

```
            if self.transform is not None:
                transformed = self.transform(image=img, mask=mask)
                return transformed['image'],transformed['mask']
            else:
                return img,mask
        else:
            if self.transform is not None:
                return self.transform(image=img)
            else:
                return img



@dataclass
class DatasetInfo:
    data_path:str
    images_folder:str
    masks_folder:str
    train_file:str
    test_file:str
    num_classes:int
    validset_ratio:float
    mean:list[float]
    std:list[float]
    class_names:list[str]


datasetInfo = DatasetInfo(
    data_path=r'C:\Users\falmasridev\Documents\opencv_courses\c2\
opencv-pytorch-segmentation-project-round2',
    images_folder=r'C:\Users\falmasridev\Documents\opencv_courses\c2\
opencv-pytorch-segmentation-project-round2\imgs\imgs',
    masks_folder=r'C:\Users\falmasridev\Documents\opencv_courses\c2\
opencv-pytorch-segmentation-project-round2\masks\masks',
    train_file=r'C:\Users\falmasridev\Documents\opencv_courses\c2\
opencv-pytorch-segmentation-project-round2\train.csv',
    test_file=r'C:\Users\falmasridev\Documents\opencv_courses\c2\
opencv-pytorch-segmentation-project-round2\test.csv',
    num_classes=12,
    validset_ratio=0.0,#since we want to compute across all samples
    mean=[0.485, 0.456, 0.406],
    std=[0.229, 0.224, 0.225],

class_names=['Background','Person','Bike','Car','Drone','Boat','Animal
','Obstacle','Construction','Vegetation','Road','Sky']

)

dataset = SemSegDataset(
    data_path=datasetInfo.data_path,
```

```
    images_folder=datasetInfo.images_folder,
    masks_folder=datasetInfo.masks_folder,
    csv_path=datasetInfo.train_file,
    dataset_type='train',
    num_classes=datasetInfo.num_classes,
    validset_ratio=datasetInfo.validset_ratio,
    transform=None,
    class_names=datasetInfo.class_names
)

def visualize_mask(image,mask):
    plt.figure(figsize=(10,10))

    rgb_mask = get_corresponding_Color(mask)

    plt.subplot(1,2,1)
    plt.title('image')
    plt.imshow(image)

    plt.subplot(1,2,2)
    plt.title('Mask')
    plt.imshow(rgb_mask)

#verify:
len(dataset)
```

```
2621
```

```
sns.set_palette("flare")
```

Q1 Answer

```
#let's calculate the image resolutions:
resolutions = []
for i in range(len(dataset)):
    image = dataset[i][0]#index 0 for image
    resolutions.append(image.shape[1] / image.shape[0])

#as we could see all images have the same resolution and that's great!
np.unique(resolutions)
```

```
array([1.77777778])
```

Q2 Answer

```
classes_frequency = {i:0 for i in range(datasetInfo.num_classes)}
for i in range(len(dataset)):
    mask = dataset[i][1]
    mask = mask.flatten()
```

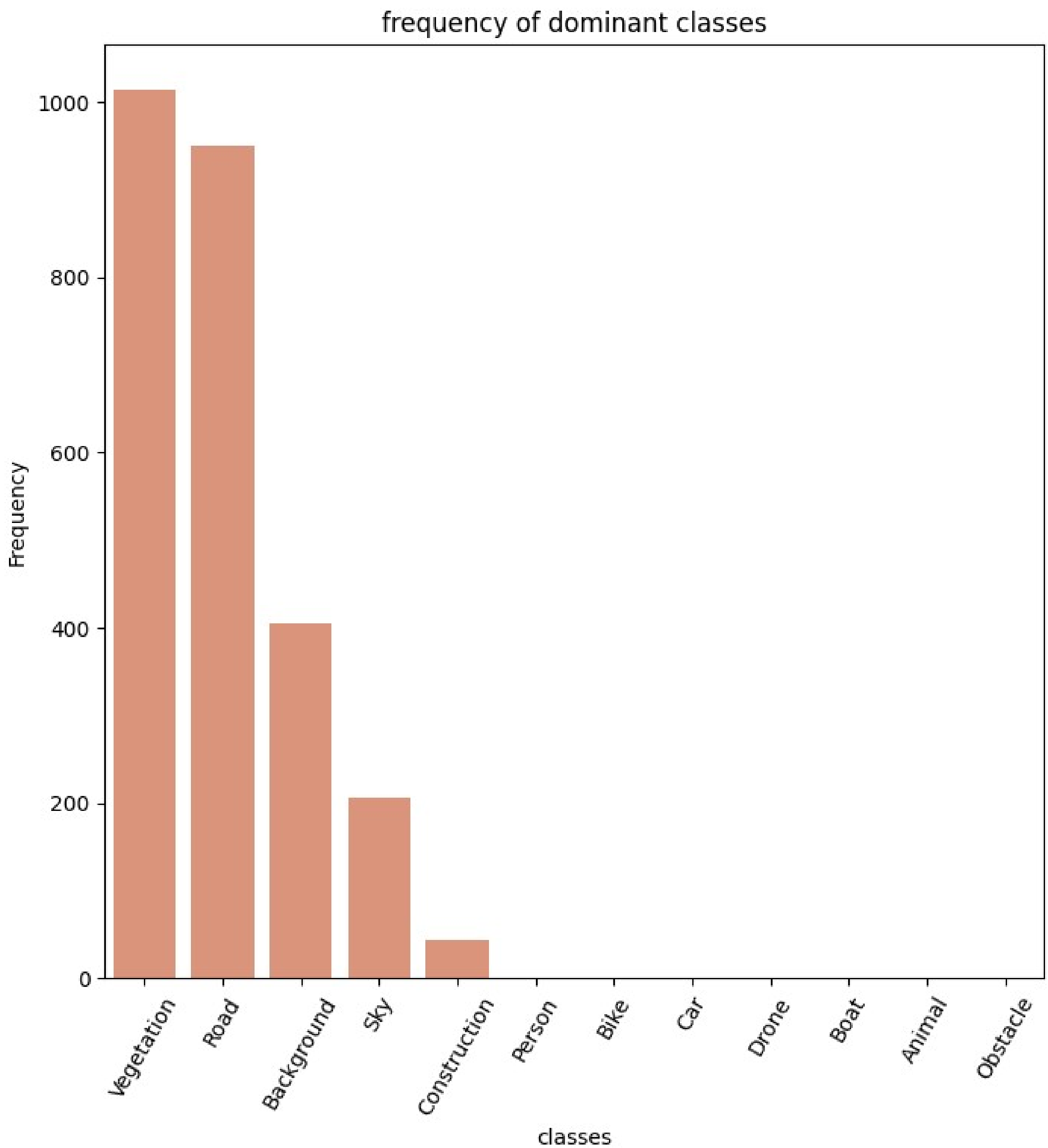
```
    classes_frequency[st.mode(mask).mode] += 1 # this will calculate
what is the most dominant class in each image

classes_frequency
df = pd.DataFrame(list(classes_frequency.items()), columns=['class',
'value']).sort_values(by='value', ascending=False)
df.reset_index(drop=True,inplace=True)
df
```

	class	value
0	9	1015
1	10	950
2	0	406
3	11	206
4	8	44
5	1	0
6	2	0
7	3	0
8	4	0
9	5	0
10	6	0
11	7	0

```
ordered_class_names = [datasetInfo.class_names[df['class'][i]] for i
in range(datasetInfo.num_classes)]
ordered_class_names
```

```
['Vegetation',
 'Road',
 'Background',
 'Sky',
 'Construction',
 'Person',
 'Bike',
 'Car',
 'Drone',
 'Boat',
 'Animal',
 'Obstacle']
```

```
ordered_class_names = [datasetInfo.class_names[df.iloc[i]['class']]
for i in range(datasetInfo.num_classes)]
plt.figure(figsize=(8,8))
sns.barplot(x=range(12),y=df['value'],data=df)
plt.xticks(range(12),ordered_class_names, rotation=60)
plt.xlabel('classes')
plt.ylabel('Frequency')
plt.title('frequency of dominant classes')
plt.savefig("most_dominant_class.png")
```



Based on that we should low the weights of these classes in order to do not let the model overfit to them.

Q3 Answer:

```
pixels_frequency = {i:
{'counted_pixels':np.longlong(0),'number_of_images':np.longlong(0)}
for i in range(datasetInfo.num_classes)}
pixels_frequency
```

```
{0: {'counted_pixels': 0, 'number_of_images': 0},
 1: {'counted_pixels': 0, 'number_of_images': 0},
 2: {'counted_pixels': 0, 'number_of_images': 0},
 3: {'counted_pixels': 0, 'number_of_images': 0},
 4: {'counted_pixels': 0, 'number_of_images': 0},
 5: {'counted_pixels': 0, 'number_of_images': 0},
 6: {'counted_pixels': 0, 'number_of_images': 0},
 7: {'counted_pixels': 0, 'number_of_images': 0},
 8: {'counted_pixels': 0, 'number_of_images': 0},
 9: {'counted_pixels': 0, 'number_of_images': 0},
 10: {'counted_pixels': 0, 'number_of_images': 0},
 11: {'counted_pixels': 0, 'number_of_images': 0}}
```

```
for i in range(len(dataset)):
    mask = dataset[i][1]
    mask = mask.flatten()
    mask = pd.Series(mask)
    result = mask.value_counts(sort=False)

    for j in result.index:
        pixels_frequency[j]['counted_pixels'] += result[j]
        pixels_frequency[j]['number_of_images'] += 1

pixels_frequency
```

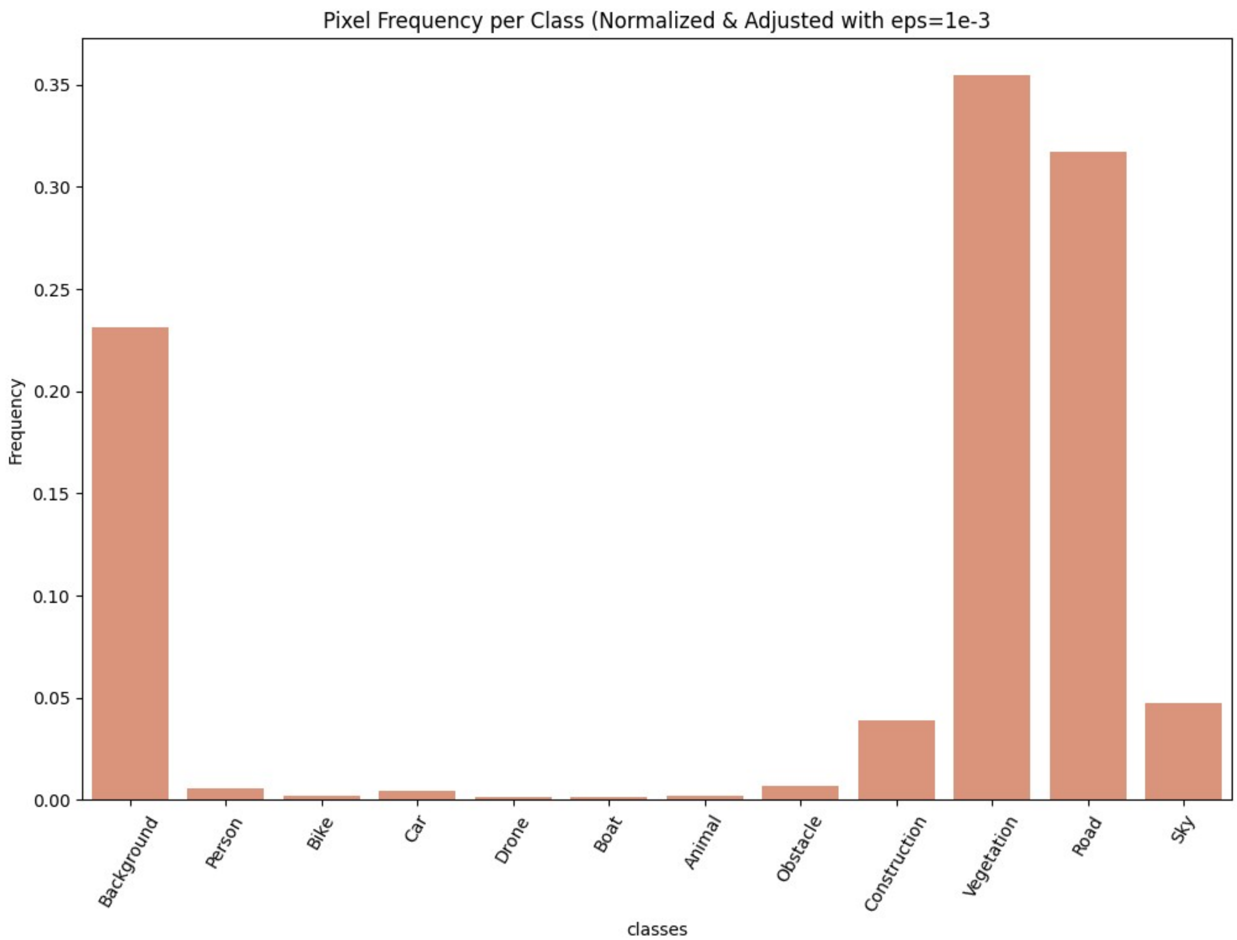
```
{0: {'counted_pixels': 555900697, 'number_of_images': 2621},
 1: {'counted_pixels': 10402499, 'number_of_images': 2223},
 2: {'counted_pixels': 1531710, 'number_of_images': 1091},
 3: {'counted_pixels': 8283476, 'number_of_images': 762},
 4: {'counted_pixels': 650353, 'number_of_images': 264},
 5: {'counted_pixels': 548987, 'number_of_images': 49},
 6: {'counted_pixels': 1663688, 'number_of_images': 73},
 7: {'counted_pixels': 14487197, 'number_of_images': 1732},
 8: {'counted_pixels': 91507497, 'number_of_images': 1455},
 9: {'counted_pixels': 854898289, 'number_of_images': 2414},
 10: {'counted_pixels': 763651425, 'number_of_images': 2294},
 11: {'counted_pixels': 111987782, 'number_of_images': 388}}
```

```
values = [pixels_frequency[i]['counted_pixels'].item() for i in
range(datasetInfo.num_classes)]
y_min, y_max = min(values), max(values)
normalized_values = [(i / sum(values)) + 1e-3 for i in values]
normalized_values
```

```
[0.23113768045023633,
 0.005306537127342193,
 0.0016341135897558184,
 0.004429281457988935,
 0.0012692400489899953,
 0.0012272754746650981,
```

```
 0.0016887512452838187,
 0.006997563830731485,
 0.038883246444979654,
 0.35491988229749566,
 0.31714453547270444,
 0.04736189255982662]
```

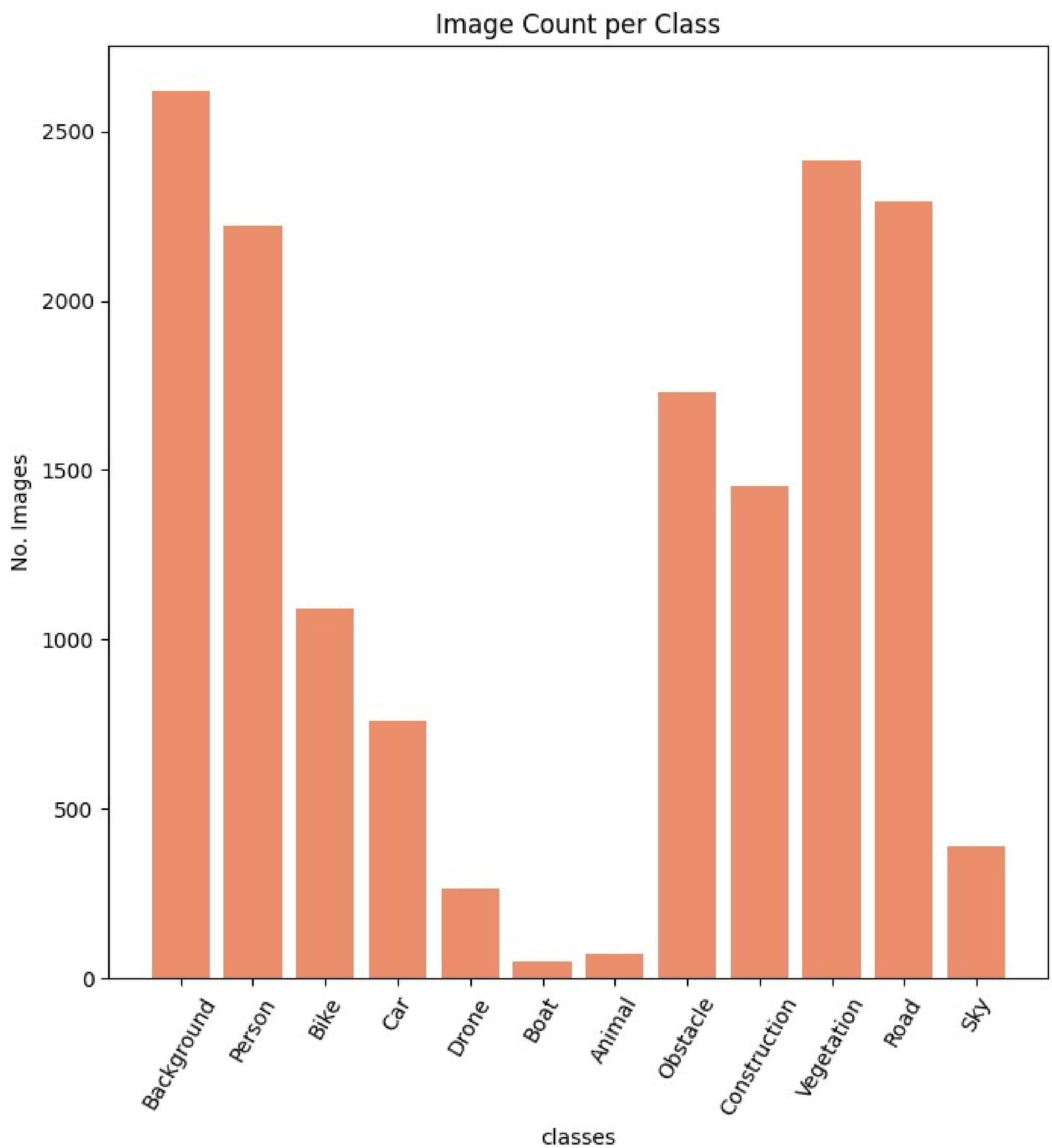
```
plt.figure(figsize=(12,8))
sns.barplot(x=range(12),y=normalized_values,log_scale=False)
plt.xticks(range(12),datasetInfo.class_names, rotation=60)
plt.xlabel('classes')
plt.ylabel('Frequency')
plt.title('Pixel Frequency per Class (Normalized & Adjusted with
eps=1e-3')
plt.savefig("pixels_frequency_per_class.png")
```




```
images_per_class = [pixels_frequency[i]['number_of_images'].item() for
i in range(datasetInfo.num_classes)]
images_per_class
```

```
[2621, 2223, 1091, 762, 264, 49, 73, 1732, 1455, 2414, 2294, 388]
```

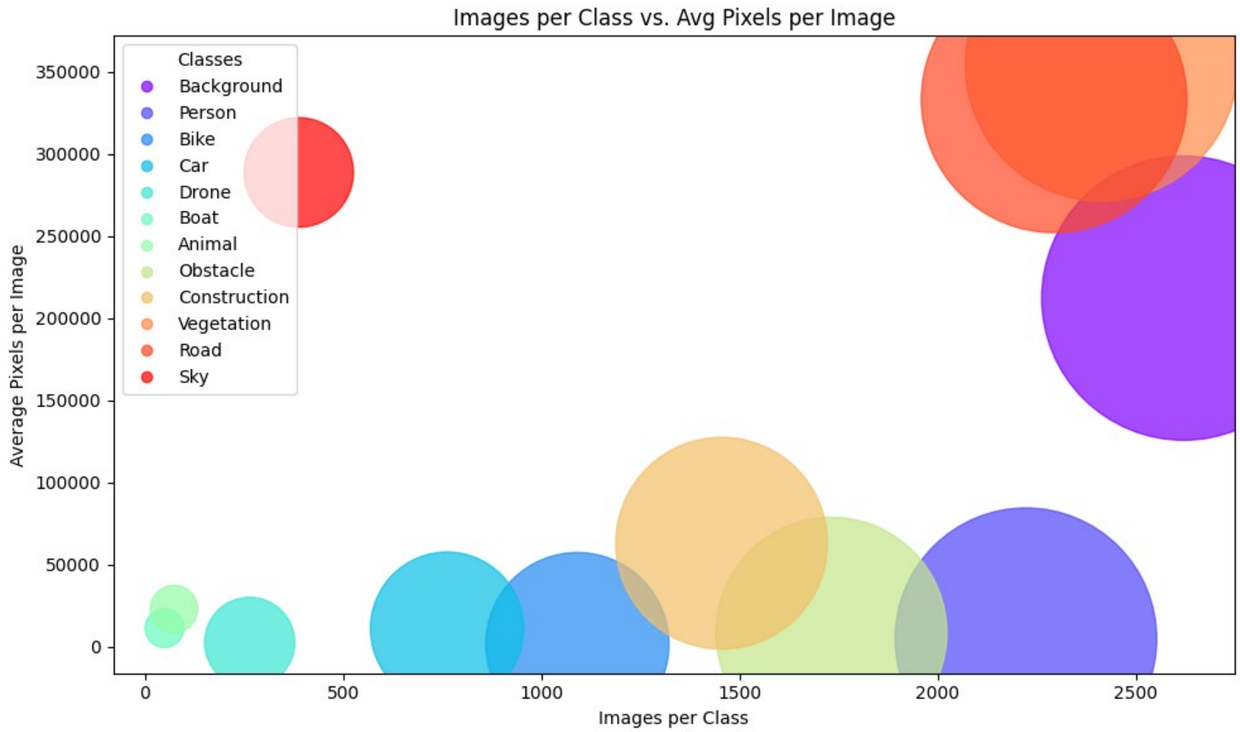
```
plt.figure(figsize=(8,8))
plt.bar(x=range(12),height=images_per_class)
plt.xticks(range(12),datasetInfo.class_names, rotation=60)
plt.xlabel('classes')
plt.ylabel('No. Images')
plt.title('Image Count per Class')
plt.savefig("image_count_per_class.png")
```



```python
avg_pixels_per_image = [pixels_frequency[i]
['counted_pixels']/pixels_frequency[i]['number_of_images'] for i in
range(datasetInfo.num_classes)]
avg_pixels_per_image
```

```
[212094.8863029378,
 4679.486729644625,
 1403.9505041246564,
 10870.703412073492,
 2463.4583333333335,
 11203.816326530612,
 22790.246575342466,
 8364.432448036952,
 62891.750515463915,
 354141.7932891466,
 332890.7693984307,
 288628.30412371136]
```

```python
plt.figure(figsize=(10, 6))
s = plt.scatter(images_per_class, avg_pixels_per_image,
                      s=np.array(images_per_class)*10,
                      c=np.arange(datasetInfo.num_classes),
                      cmap='rainbow', alpha=0.7)
plt.legend(handles=s.legend_elements()[0],
labels=datasetInfo.class_names, title="Classes", loc='upper left',
bbox_to_anchor=(0, 1))
plt.xlabel('Images per Class')
plt.ylabel('Average Pixels per Image')
plt.title('Images per Class vs. Avg Pixels per Image')
plt.tight_layout()
plt.savefig("Images_per_Class_vs_Avg_Pixels_per_Image.png")
```



avg objects location using heatmap

```
heatmap_dict = {i:np.zeros((720,1280)) for i in
range(datasetInfo.num_classes)}
for c_index in range(datasetInfo.num_classes):
    for i in range(len(dataset)):
        mask = dataset[i][1]
        heatmap_dict[c_index] += mask == c_index
```

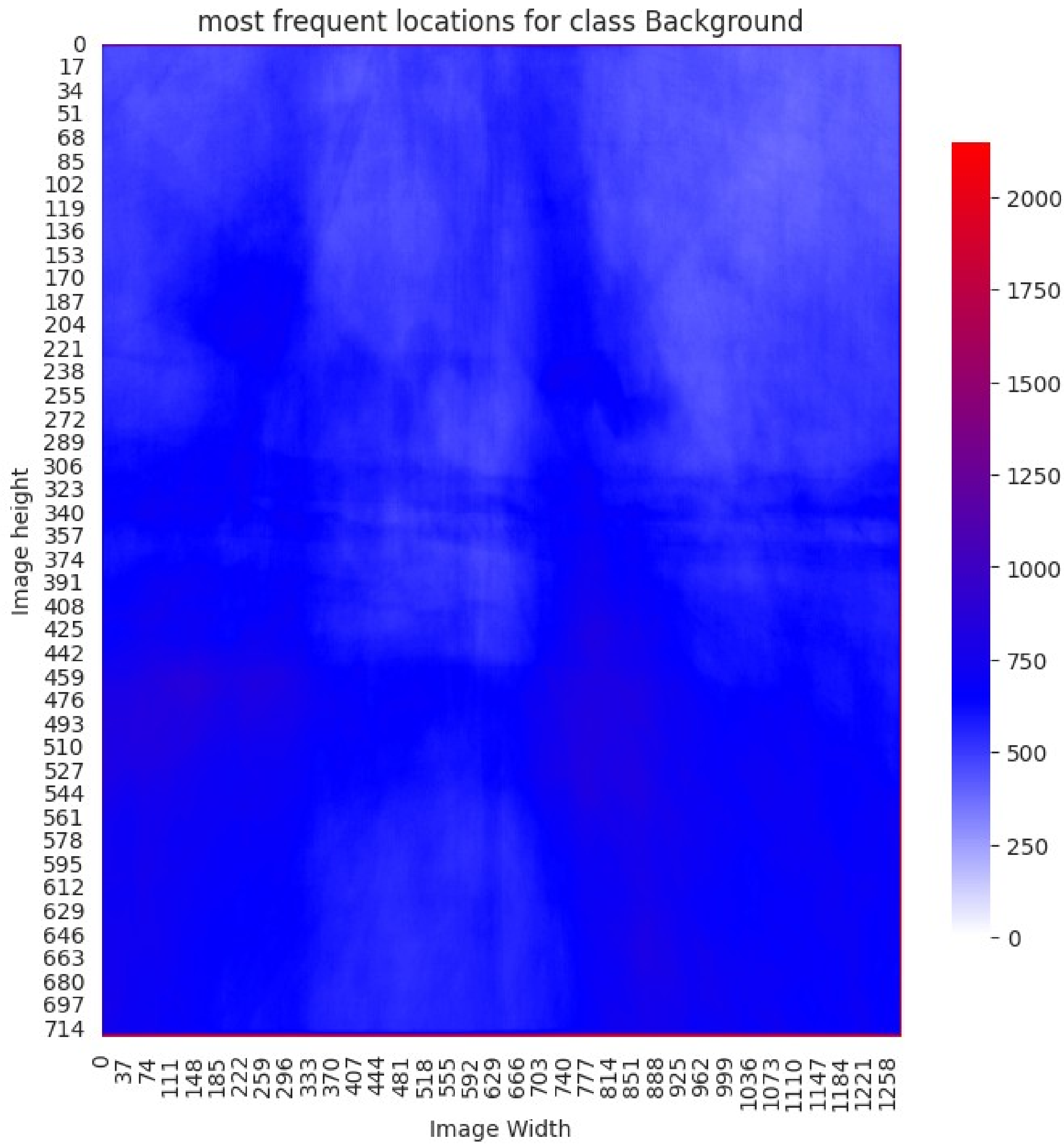
```
heatmap_dict
```

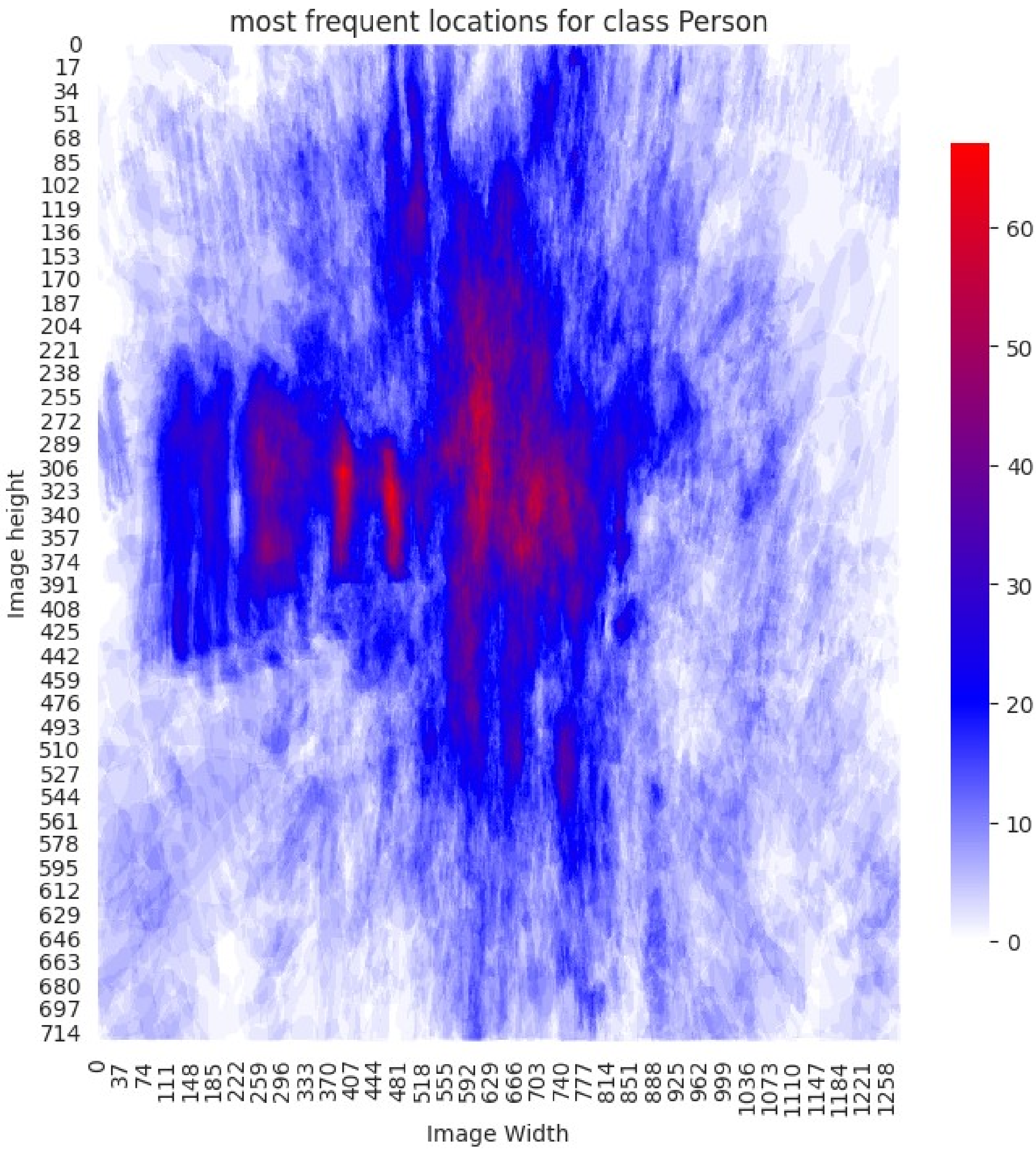
```
{0: array([[1834., 1609., 1576., ..., 1571., 1635., 1977.],
        [1548.,  979.,  870., ...,  864., 1043., 1726.],
        [1437.,  731.,  583., ...,  563.,  830., 1671.],
        ...,
        [1710., 1061.,  881., ...,  817., 1026., 1771.],
        [1827., 1322., 1205., ..., 1150., 1279., 1856.],
        [2130., 1985., 1971., ..., 1919., 1937., 2110.]]),
 1: array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 1., 1., 1.],
        [0., 0., 0., ..., 1., 1., 1.],
        [0., 0., 0., ..., 1., 1., 1.]]),
 2: array([[1., 1., 1., ..., 0., 0., 0.],
```

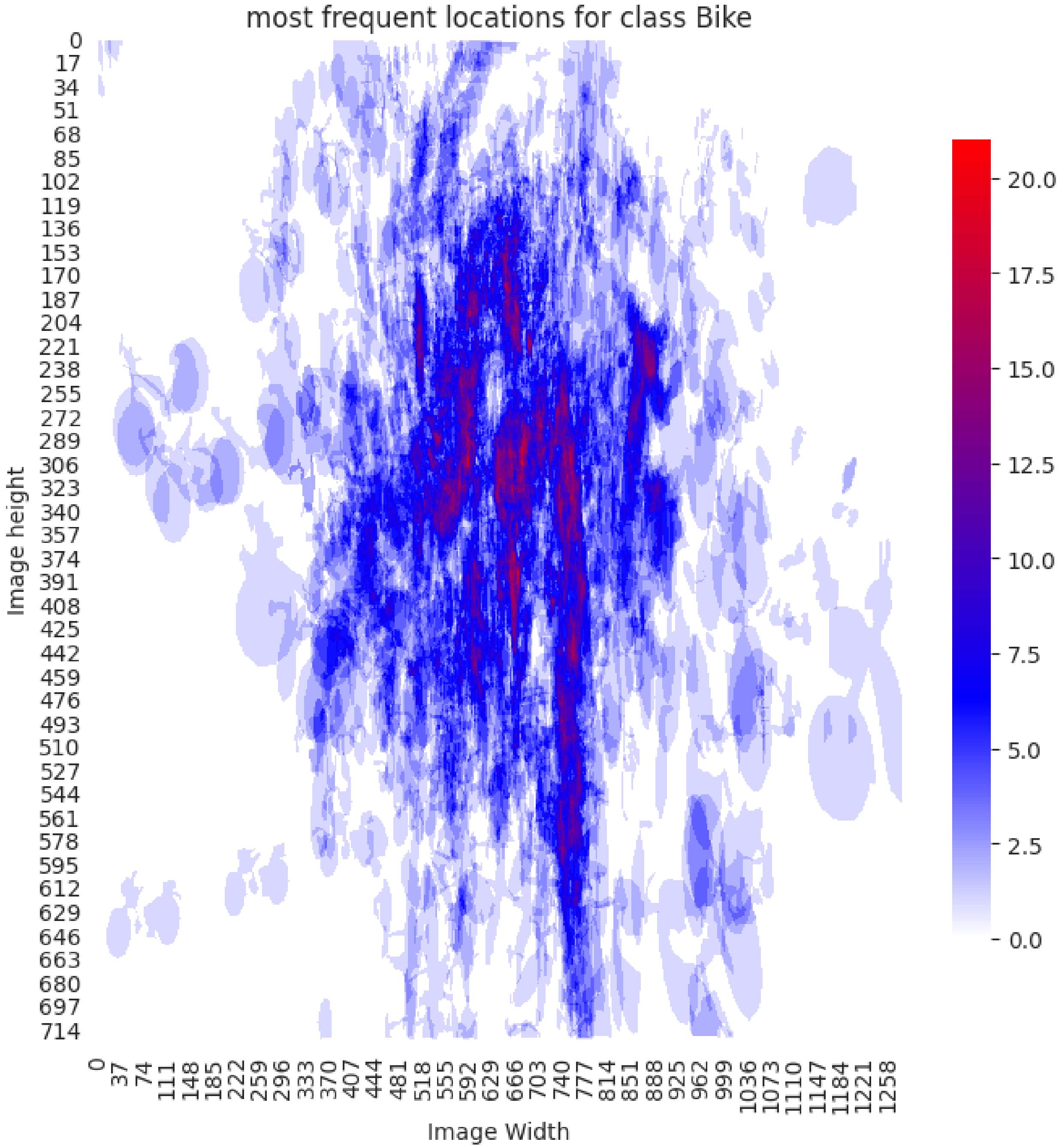
```
        [1., 1., 1., ..., 0., 0., 0.],
        [1., 1., 1., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 3: array([[10., 11., 11., ...,  0.,  0.,  0.],
        [10., 12., 13., ...,  1.,  1.,  0.],
        [10., 12., 13., ...,  1.,  1.,  0.],
        ...,
        [ 1.,  2.,  2., ...,  2.,  2.,  1.],
        [ 1.,  2.,  2., ...,  2.,  2.,  1.],
        [ 0.,  1.,  1., ...,  1.,  1.,  1.]]),
 4: array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 5: array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 6: array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]),
 7: array([[13., 15., 16., ...,  5.,  5.,  5.],
        [17., 23., 25., ...,  7.,  5.,  5.],
        [18., 25., 26., ...,  8.,  6.,  5.],
        ...,
        [11., 12., 13., ..., 10., 10.,  5.],
        [11., 12., 12., ..., 10., 10.,  5.],
        [10., 10.,  9., ...,  7.,  7.,  4.]]),
 8: array([[ 60., 104., 112., ..., 174., 167., 105.],
        [103., 210., 226., ..., 275., 256., 150.],
        [116., 233., 253., ..., 304., 276., 154.],
        ...,
        [ 44.,  89., 101., ..., 114., 107.,  65.],
        [ 39.,  74.,  82., ...,  99.,  95.,  64.],
        [ 18.,  28.,  29., ...,  57.,  56.,  41.]]),
 9: array([[ 472.,  595.,  611., ...,  587.,  547.,  344.],
```

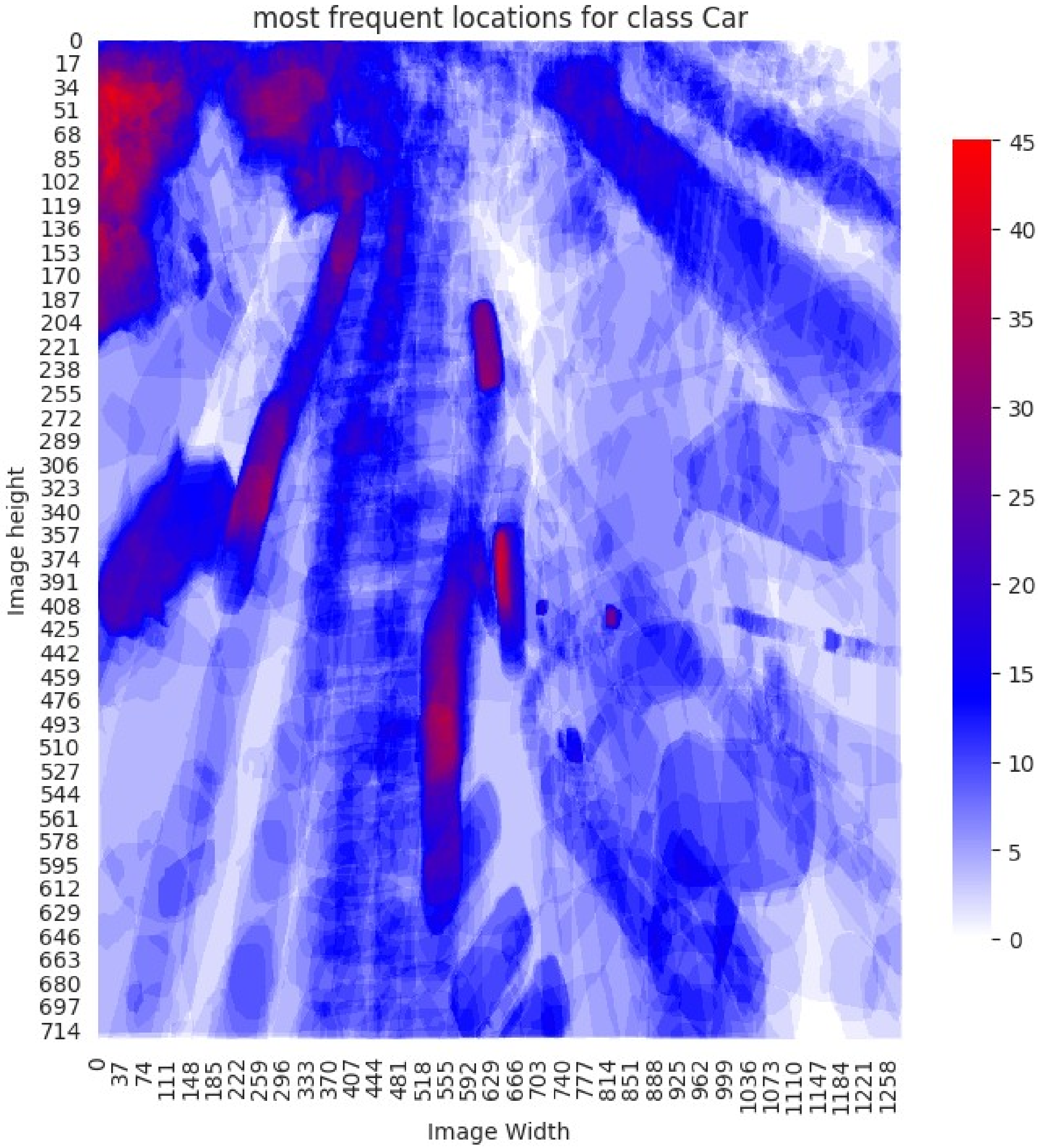
```
        [ 619.,  931.,  988., ...,  953.,  850.,  458.],
        [ 683., 1089., 1172., ..., 1168.,  995.,  494.],
        ...,
        [ 454.,  797.,  915., ..., 1150., 1040.,  582.],
        [ 399.,  664.,  736., ...,  950.,  880.,  521.],
        [ 230.,  311.,  321., ...,  479.,  467.,  356.]]),
 10: array([[ 95., 122., 128., ..., 133., 119.,  74.],
        [145., 202., 225., ..., 214., 180.,  95.],
        [157., 231., 258., ..., 250., 205., 103.],
        ...,
        [401., 660., 709., ..., 527., 435., 196.],
        [344., 547., 584., ..., 409., 354., 173.],
        [233., 286., 290., ..., 157., 152., 108.]]),
 11: array([[136., 164., 166., ..., 151., 148., 116.],
        [178., 263., 273., ..., 307., 286., 187.],
        [199., 299., 315., ..., 327., 308., 194.],
        ...,
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.]])}
```

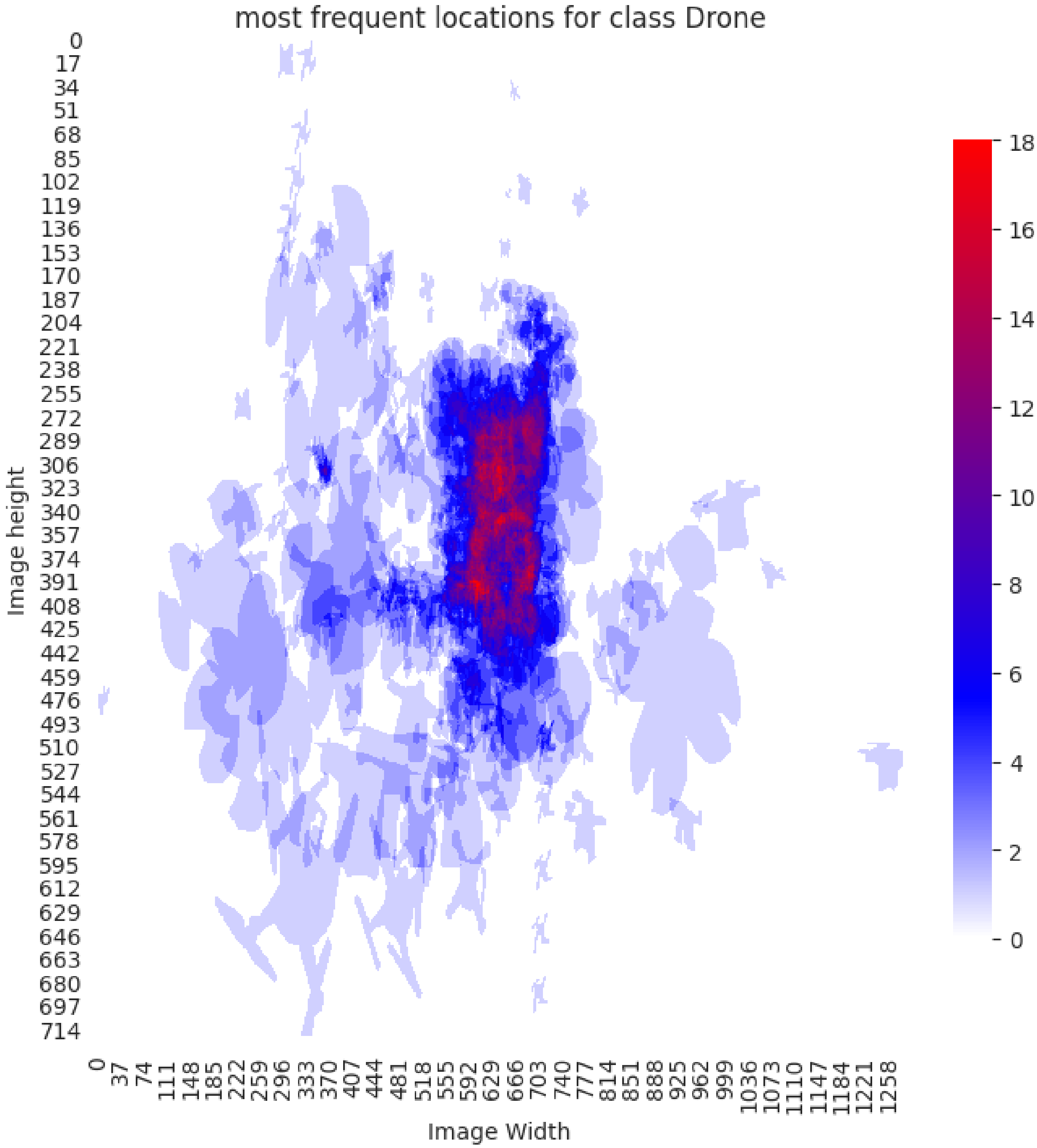
```
cmap = LinearSegmentedColormap.from_list(
    "custom_cmap", [(0, 'white'), (0.3, 'blue'), (1, 'red')]
)

for i in range(datasetInfo.num_classes):
    with sns.axes_style("white"):
        plt.figure(figsize=(8,8))
        sns.heatmap(heatmap_dict[i],cmap=cmap,cbar_kws={'shrink':
0.8}, vmin=0, vmax=np.max(heatmap_dict[i]))
        plt.title(f"most frequent locations for class
{datasetInfo.class_names[i]}")
        plt.xlabel("Image Width")
        plt.ylabel("Image height")
        plt.savefig(f"{datasetInfo.class_names[i]}_heatmap.png",)
```

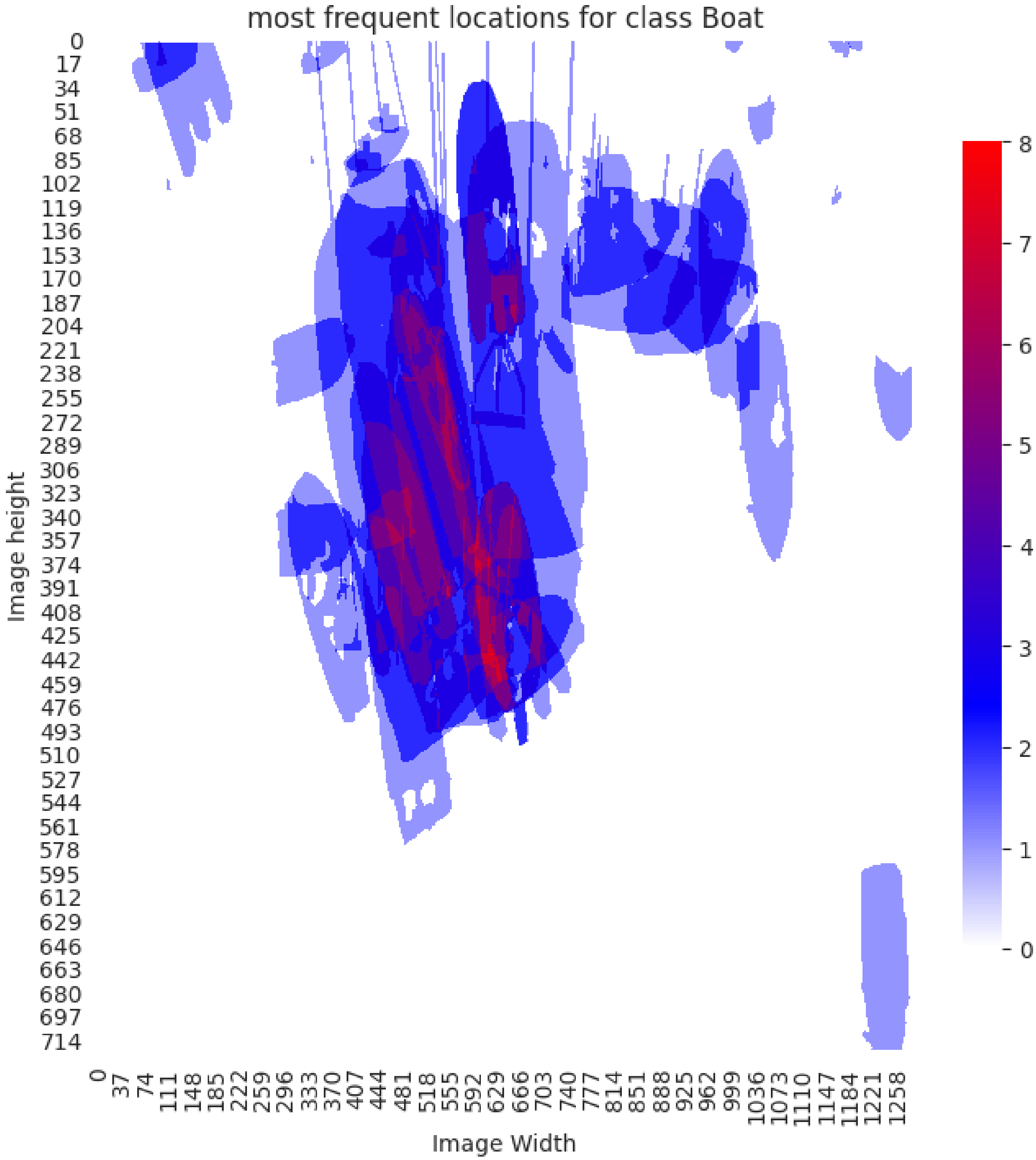

most frequent locations for class Background
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
2000
1750
1500
1250
1000
750
500
250
0

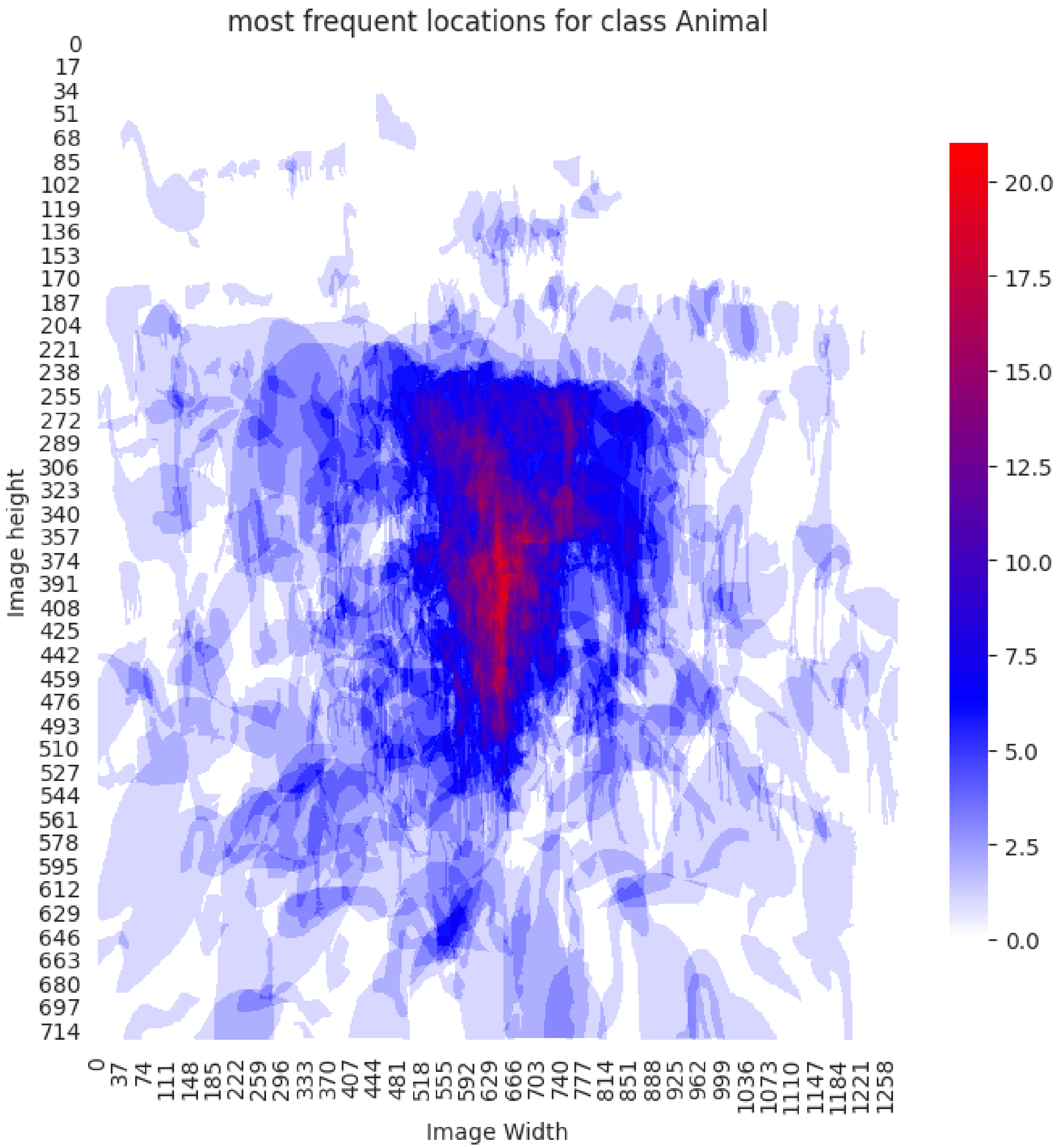

most frequent locations for class Person
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
0
10
20
30
40
50
60

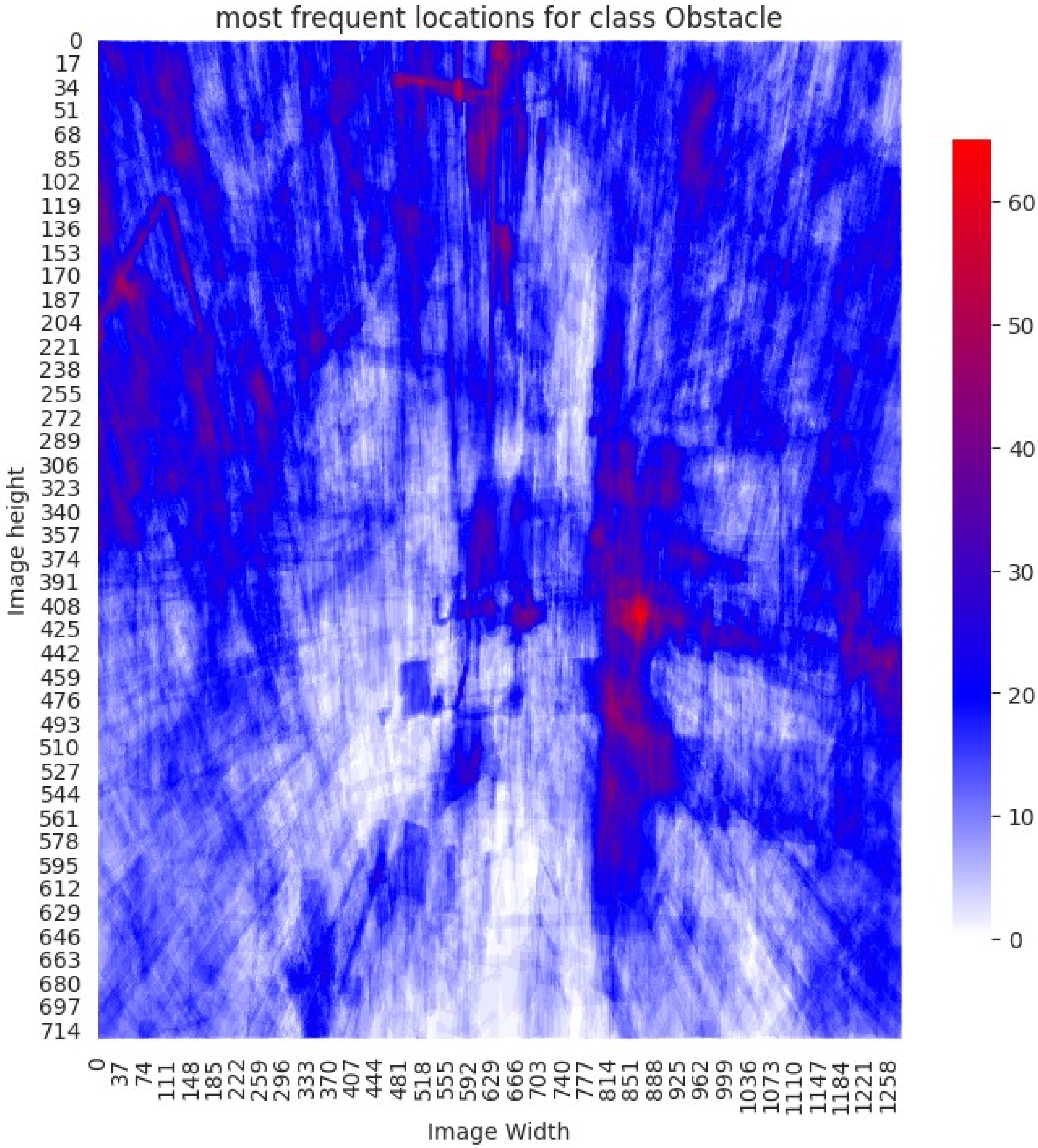

most frequent locations for class Bike
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0


most frequent locations for class Car
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
45
40
35
30
25
20
15
10
5
0

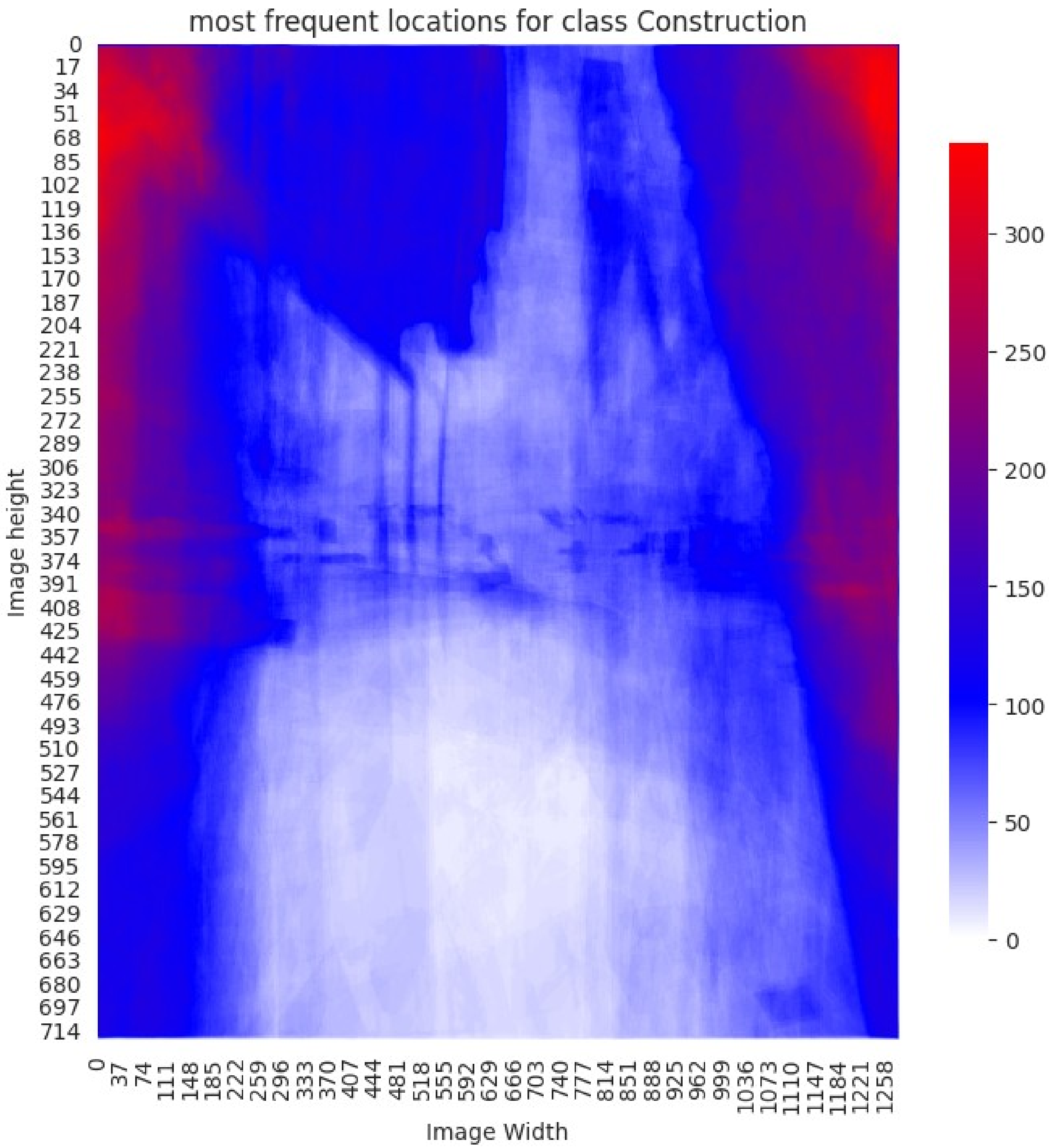

most frequent locations for class Drone
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
18
16
14
12
10
8
6
4
2
0

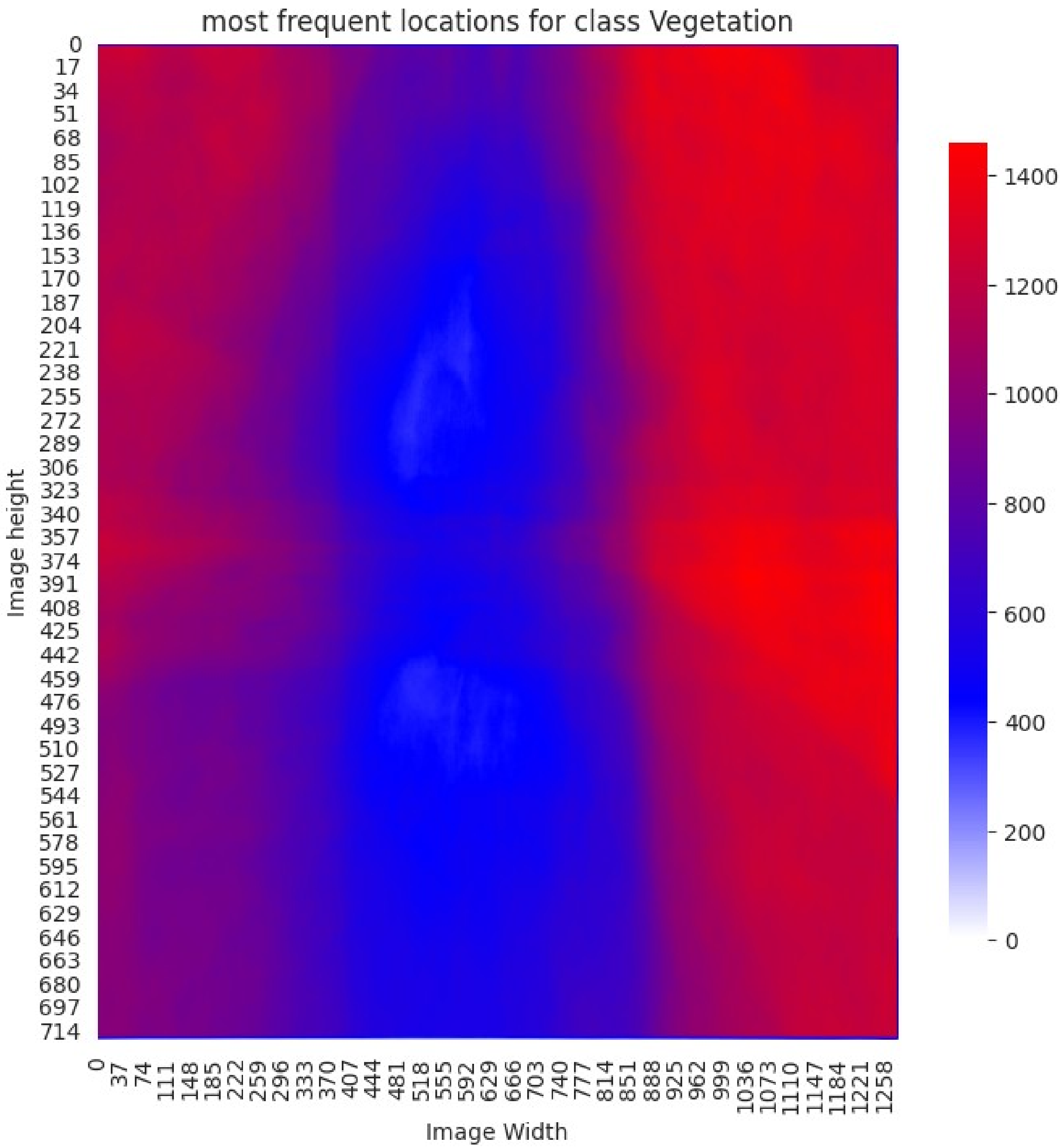

most frequent locations for class Boat
Image height
Image Width
0 17 34 51 68 85 102 119 136 153 170 187 204 221 238 255 272 289 306 323 340 357 374 391 408 425 442 459 476 493 510 527 544 561 578 595 612 629 646 663 680 697 714
0 37 74 111 148 185 222 259 296 333 370 407 444 481 518 555 592 629 666 703 740 777 814 851 888 925 962 999 1036 1073 1110 1147 1184 1221 1258
0 1 2 3 4 5 6 7 8

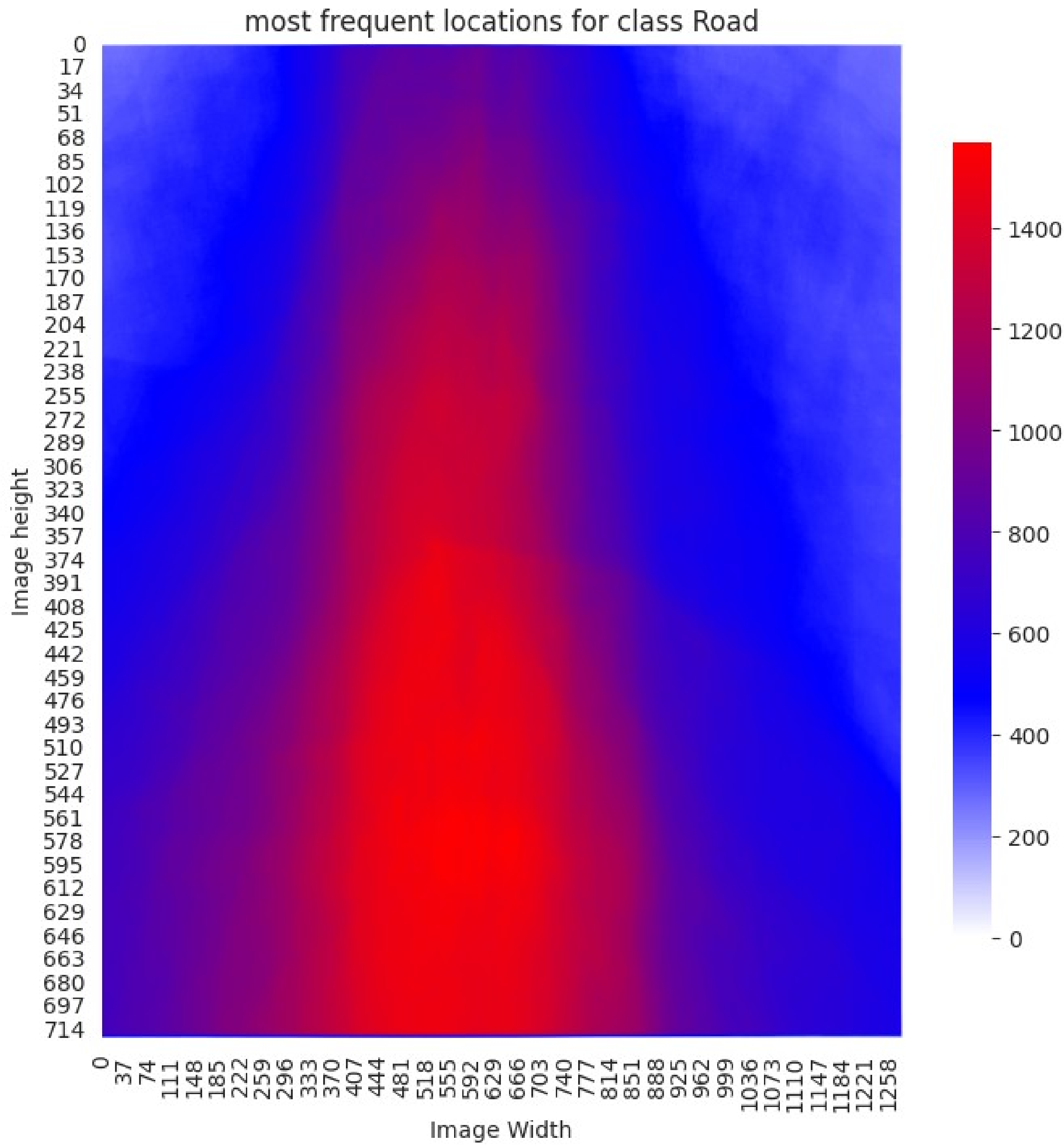

most frequent locations for class Animal
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0



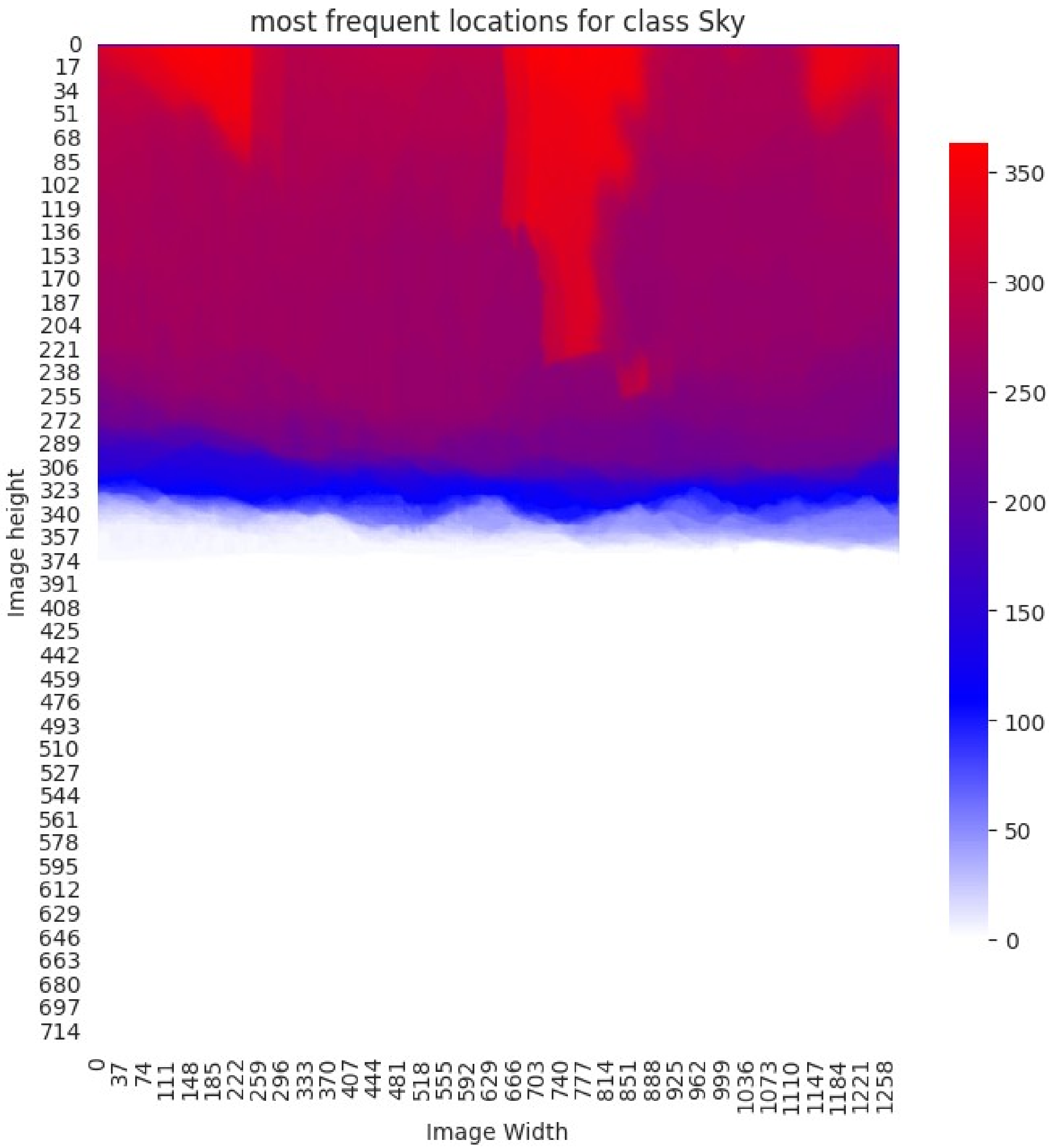
most frequent locations for class Obstacle
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
60
50
40
30
20
10
0


most frequent locations for class Construction
Image height
Image Width
0
50
100
150
200
250
300


most frequent locations for class Vegetation
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
1400
1200
1000
800
600
400
200
0



most frequent locations for class Road
Image height
Image Width
0
17
34
51
68
85
102
119
136
153
170
187
204
221
238
255
272
289
306
323
340
357
374
391
408
425
442
459
476
493
510
527
544
561
578
595
612
629
646
663
680
697
714
0
37
74
111
148
185
222
259
296
333
370
407
444
481
518
555
592
629
666
703
740
777
814
851
888
925
962
999
1036
1073
1110
1147
1184
1221
1258
1400
1200
1000
800
600
400
200
0




```
visualize_mask(*dataset[1962])
plt.savefig("sample.png")
```

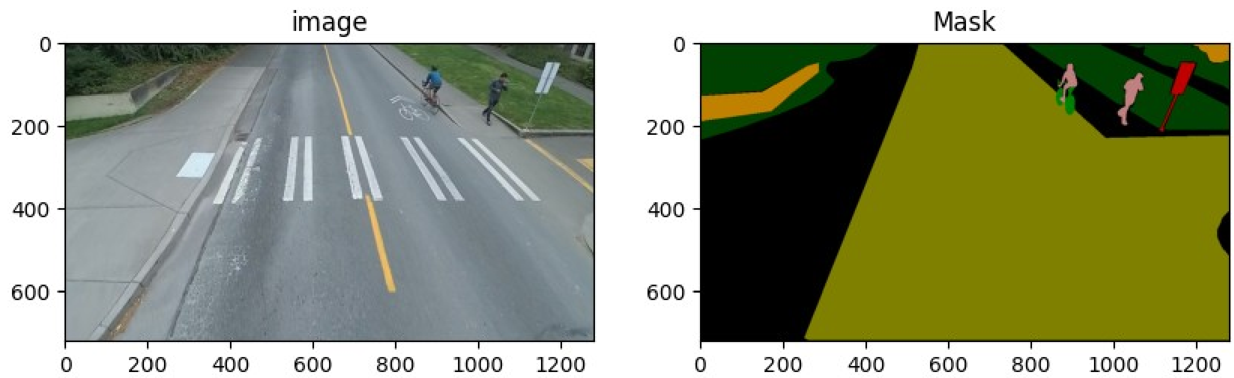


image
Mask
0
200
400
600
0
200
400
600
800
1000
1200