Transfer learning based damage road detection for multiple countries

Task :

1. Object Detection and classification.
2. Fine tune pretrained models to our dataset.

Link for referenence for training models:

https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html (https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html)

In [1]:

```
!pip install imgaug
```

```
Requirement already satisfied: imgaug in f:\anaconda3\lib\site-packages (0.4.0)
Requirement already satisfied: six in f:\anaconda3\lib\site-packages (from imgaug) (1.16.0)
Requirement already satisfied: Pillow in f:\anaconda3\lib\site-packages (from imgaug) (9.4.0)
Requirement already satisfied: Shapely in f:\anaconda3\lib\site-packages (from imgaug) (2.0.1)
Requirement already satisfied: scipy in f:\anaconda3\lib\site-packages (from imgaug) (1.10.0)
Requirement already satisfied: numpy>=1.15 in f:\anaconda3\lib\site-packages (from imgaug) (1.23.5)
Requirement already satisfied: opencv-python in f:\anaconda3\lib\site-packages (from imgaug) (4.7.0.72)
Requirement already satisfied: imageio in f:\anaconda3\lib\site-packages (from imgaug) (2.26.0)
Requirement already satisfied: scikit-image>=0.14.2 in f:\anaconda3\lib\site-packages (from imgaug) (0.19.3)
Requirement already satisfied: matplotlib in f:\anaconda3\lib\site-packages (from imgaug) (3.7.0)
Requirement already satisfied: PyWavelets>=1.1.1 in f:\anaconda3\lib\site-packages (from scikit-image>=0.14.2->imgaug) (1.
4.1)
Requirement already satisfied: tifffile>=2019.7.26 in f:\anaconda3\lib\site-packages (from scikit-image>=0.14.2->imgaug) (2
021.7.2)
Requirement already satisfied: packaging>=20.0 in f:\anaconda3\lib\site-packages (from scikit-image>=0.14.2->imgaug) (22.0)
Requirement already satisfied: networkx>=2.2 in f:\anaconda3\lib\site-packages (from scikit-image>=0.14.2->imgaug) (2.8.4)
Requirement already satisfied: kiwisolver>=1.0.1 in f:\anaconda3\lib\site-packages (from matplotlib->imgaug) (1.4.4)
Requirement already satisfied: contourpy>=1.0.1 in f:\anaconda3\lib\site-packages (from matplotlib->imgaug) (1.0.5)
Requirement already satisfied: pyparsing>=2.3.1 in f:\anaconda3\lib\site-packages (from matplotlib->imgaug) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in f:\anaconda3\lib\site-packages (from matplotlib->imgaug) (4.25.0)
Requirement already satisfied: cycler>=0.10 in f:\anaconda3\lib\site-packages (from matplotlib->imgaug) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in f:\anaconda3\lib\site-packages (from matplotlib->imgaug) (2.8.2)
```

In [2]:

```
import os
from xml.etree import ElementTree
from bs4 import BeautifulSoup
import seaborn as sns
import collections
import pandas as pd
import xml.etree.ElementTree as ET
import cv2
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import shutil
from imgaug.augmentables.bbs import BoundingBox, BoundingBoxesOnImage
from imgaug import augmenters as iaa
import imgaug as ia
import imageio
import re
import numpy as np
```

In [3]:

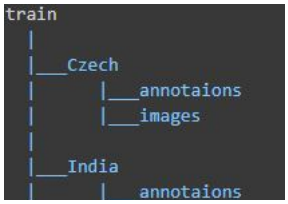
```
#!wget --header="Host: doc-0k-98-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/53
```

In [4]:

```
#untar train data
#!tar -xvf /content/drive/MyDrive/object_detection/train.tar
```

after unzipping:

we get following folders



We can see We have data for three countries namely **Czech**,**India** and **Japan**.

In each country folder we have annotation folder and images folder.

annotation folder contains xml filesin PASCAL VOC format. We will see what exactly data xml file have.

image folder contains images in jpg format.

label_map.pbtxt file have labels with thier id's. We will see that as well

First look at Xml file

In [5]:

```
%cd \rdd080423\Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main
```

```
F:\rdd080423\Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main
```

In [6]:

```
import os
cwd = os.getcwd()
print(cwd)
```

```
F:\rdd080423\Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main
```

In [7]:

```
#We will check content in .xml file given for image


with open('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/train/Czech/annotations/xmls/Cze
    data = f.read()

Bs_data = BeautifulSoup(data, "xml")
Bs_data
```

Out[7]:

```
<?xml version="1.0" encoding="utf-8"?>
<annotation>
<folder>images</folder>
<filename>Czech_000010.jpg</filename>
<size>
<depth>3</depth>
<width>600</width>
<height>600</height>
</size>
<object>
<name>D40</name>
<bndbox>
<xmin>213</xmin>
<ymin>409</ymin>
<xmax>274</xmax>
<ymax>441</ymax>
</bndbox>
</object>
<object>
<name>D10</name>
<bndbox>
<xmin>228</xmin>
<ymin>473</ymin>
<xmax>327</xmax>
<ymax>495</ymax>
</bndbox>
</object>
</annotation>
```

In [8]:

```
%cd content
```

```
F:\rdd080423\Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main\content
```

From xml file we can see following thing

1. Image name

2. image size
3. object with x,y axis dimension
4. In single image we can have multiple objects that we have to class.

Now we will see total images including all countries and labels

In [9]:

```
# the number of total images and total labels.
country = ['Czech','India','Japan']
base_path = os.getcwd() + '/train/'
cls_names = []
total_images = 0
for cont in country:

    file_list = [filename for filename in os.listdir(base_path + cont + '/annotations/xmls/')]

    for file in file_list:

        total_images = total_images + 1

        infile_xml = open(base_path + cont + '/annotations/xmls/' +file)
        tree = ElementTree.parse(infile_xml)
        root = tree.getroot()
        for obj in root.iter('object'):
            cls_name = obj.find('name').text
            cls_names.append(cls_name)

print("Total number of images: " + str(total_images))
print("Total number of labels: " + str(len(cls_names)))
```

```
Total number of images: 21041
Total number of labels: 34702
```

In [10]:

```
#check classes
damageTypes=list(set(cls_names))
print(damageTypes)
print('number of classes: ', len(damageTypes))
```

```
['D11', 'D10', 'D50', 'D00', 'D20', 'D0w0', 'D43', 'D40', 'D44', 'D01']
number of classes:  10
```

So, We have total 10 classes.

Now we will see distribution of these classes

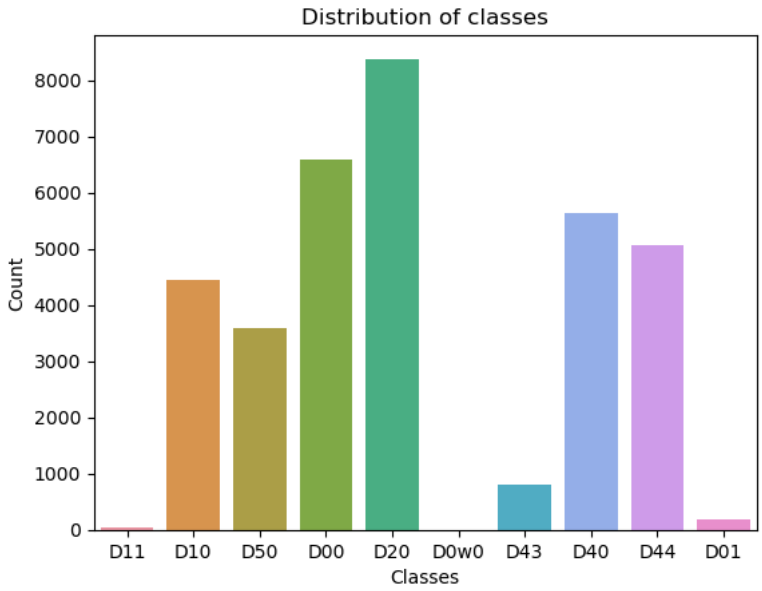
In [11]:

```
# We will see Classes distribution for whole data
count_dict = collections.Counter(cls_names)
cls_count = []
for damageType in damageTypes:
    print(str(damageType) + ' : ' + str(count_dict[damageType]))
    cls_count.append(count_dict[damageType])

sns.set_palette("winter", 8)
sns.barplot(x=damageTypes,  y=cls_count)
plt.xlabel('Classes')
plt.ylabel('Count')
plt.title('Distribution of classes')
```

```
D11 : 45
D10 : 4446
D50 : 3581
D00 : 6592
D20 : 8381
D0w0 : 1
D43 : 793
D40 : 5627
D44 : 5057
D01 : 179
```

Out[11]:

```
Text(0.5, 1.0, 'Distribution of classes')
```



We can see from above distribution that classes are not balance.

D0w0, D11, D01 and D43 are very rare classes.

Now we will see countriwise images count and label count

In [12]:

```
# the number of each class labels for each country
for gov in country:
    cls_names = []
    total_images = 0
    file_list = [filename for filename in os.listdir(base_path + gov + '/annotations/xmls/')]

    for file in file_list:

        total_images = total_images + 1

        infile_xml = open(base_path + gov + '/annotations/xmls/' +file)
        tree = ElementTree.parse(infile_xml)
        root = tree.getroot()
        for obj in root.iter('object'):
          cls_name = obj.find('name').text
          cls_names.append(cls_name)
    print('Country : ',gov)
    print("\nTotal images in {} :  {}".format(gov , total_images))
    print("Total labels in {} :  {}\n".format(gov , len(cls_names)))

    count_dict = collections.Counter(cls_names)
    cls_count = []
    for damageType in damageTypes:
        print(str(damageType) + ' : ' + str(count_dict[damageType]))
        cls_count.append(count_dict[damageType])
    print('**********************************************************************************\n')
```

```
Country :  Czech

Total images in Czech :  2829
Total labels in Czech :  1745

D11 : 0
D10 : 399
D50 : 0
D00 : 988
D20 : 161
D0w0 : 0
D43 : 0
D40 : 197
D44 : 0
D01 : 0
**********************************************************************************

Country :  India

Total images in India :  7706
Total labels in India :  8203

D11 : 45
D10 : 68
D50 : 28
D00 : 1555
D20 : 2021
D0w0 : 1
D43 : 57
D40 : 3187
D44 : 1062
D01 : 179
**********************************************************************************

Country :  Japan

Total images in Japan :  10506
Total labels in Japan :  24754

D11 : 0
D10 : 3979
D50 : 3553
D00 : 4049
D20 : 6199
D0w0 : 0
D43 : 736
D40 : 2243
D44 : 3995
D01 : 0
**********************************************************************************
```

1. Czech country have only four classes (label)
2. India have all classes (10 classes)
3. Japan have 7 classes.

So we can say that only india contains all classes. Also in india we have class 'D0w0' occurs only once. We can remove that or wil not include while training.

Japan have more numbere of images compare to Czech and India.

So we have more versetile data.

We will create two folder that will contain all the images and annotations file.

In [13]:

```
! mkdir data
!dir
```

```
A subdirectory or file data already exists.

 Volume in drive F is New Volume
 Volume Serial Number is E419-B038

 Directory of F:\rdd080423\Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main\content

08-04-2023  15:04    <DIR>          .
08-04-2023  15:27    <DIR>          ..
08-04-2023  15:26    <DIR>          data
12-05-2020  05:09               127 label_map.pbtxt
17-05-2020  15:57    <DIR>          test1
08-04-2023  13:07       186,476,694 test1.tar.gz
12-05-2020  05:09    <DIR>          train
08-04-2023  13:20     1,472,626,254 train.tar.gz
               3 File(s)  1,659,103,075 bytes
               5 Dir(s)  67,859,415,040 bytes free
```

In [14]:

```
base_path = 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/train/'
for cont in country:
  for fil in os.listdir(base_path + cont + '/images/'):
    shutil.copy(base_path + cont + '/images/'+ fil, "f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-mai
file_list = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/c
print('Total images in Image folder  : ',len(file_list))
```

```
Total images in Image folder  :  19881
```

In [15]:

```
base_path = 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/train/'
for cont in country:
  for fil in os.listdir(base_path + cont + '/annotations/xmls'):
    shutil.copy(base_path + cont + '/annotations/xmls/'+ fil, "f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Cou

file_list = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/c
print('Total images in Annotation folder  : ',len(file_list))
```

```
Total images in Annotation folder  :  21041
```

Convert xml to csv

In [16]:

```
def xml_to_csv(base_path):
    xml_list = []
    file_list = [filename for filename in os.listdir(base_path)]
    for file in file_list:
          infile_xml = open('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Annotatio
          #print(base_path + gov + '/annotations/xmls/' + file)
          tree = ElementTree.parse(infile_xml)
          root = tree.getroot()
          for obj in root.findall('object'):
              value = (str(root.find('filename').text),
                       int(root.find('size').find('width').text),
                       int(root.find('size').find('height').text),
                       obj.find('name').text,
                       int(obj.find('bndbox').find('xmin').text),
                       int(obj.find('bndbox').find('ymin').text),
                       int(obj.find('bndbox').find('xmax').text),
                       int(obj.find('bndbox').find('ymax').text))
              xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'label', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


base_path =  'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images'
labels_df = xml_to_csv(base_path)
labels_df.head(10)
```

Out[16]:

	filename	width	height	label	xmin	ymin	xmax	ymax
0	Czech_000041.jpg	600	600	D20	359	489	479	527
1	Czech_000047.jpg	600	600	D00	52	415	181	510
2	Czech_000054.jpg	600	600	D00	29	370	235	517
3	Czech_000055.jpg	600	600	D00	147	408	201	448
4	Czech_000055.jpg	600	600	D10	329	489	460	519
5	Czech_000061.jpg	600	600	D00	249	466	269	518
6	Czech_000062.jpg	600	600	D00	303	372	505	518
7	Czech_000062.jpg	600	600	D00	1	400	203	526
8	Czech_000064.jpg	600	600	D40	81	409	134	425
9	Czech_000067.jpg	600	600	D00	366	458	417	542

In [17]:

```
labels_df.shape
```

Out[17]:

```
(31880, 8)
```

In [18]:

```
image_with_label = list(set(labels_df.filename.values))

print('Number of images have labels : ', len(image_with_label))
```

```
Number of images have labels :  13487
```

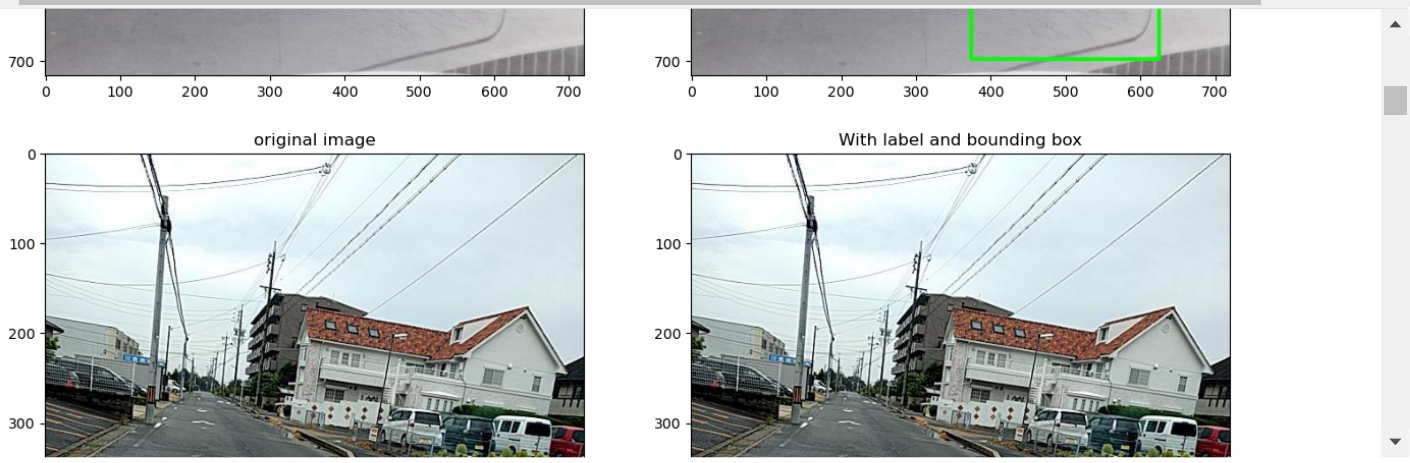
In [19]:

```
ename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images')]
filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Annotations')]
```

Plotting images with labels and bounding box

In [20]:

```
def plot_image (img_file):
  gov = img_file.split('.')[0]
  image = cv2.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/' + gov +
  image1 = cv2.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/' + gov

  infile_xml = open('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Annotations/' + g
  tree = ElementTree.parse(infile_xml)
  root = tree.getroot()

  for obj in root.iter('object'):
        cls_name = obj.find('name').text
        xmlbox = obj.find('bndbox')
        xmin = int(xmlbox.find('xmin').text)
        xmax = int(xmlbox.find('xmax').text)
        ymin = int(xmlbox.find('ymin').text)
        ymax = int(xmlbox.find('ymax').text)

        font = cv2.FONT_HERSHEY_SIMPLEX

        # put text
        image1 = cv2.putText(image1,cls_name,(xmin,ymin-10),font,1,(0,255,0),2,cv2.LINE_AA)

        # draw bounding box
        image1 = cv2.rectangle(image1, (xmin, ymin), (xmax, ymax), (0,255,0),3)
  plt.figure(figsize=(15,15))
  plt.axis('off')
  plt.subplot(121)
  plt.title('original image')
  plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
  plt.subplot(122)
  plt.title('With label and bounding box')
  plt.imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))
  plt.show()

image_with_label = list(set(labels_df.filename.values))
for i in range(0,15):
  plot_image(image_with_label [i])
```

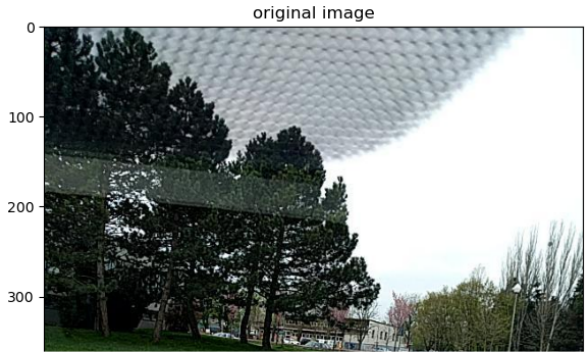


Potting some images and images with bounding box with class from csv file

In [21]:

```
ow_image_objects(image_row):
_path = 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/'
path = 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/' + image_row.filename

e = image_row.label
e = cv2.imread(img_path)
e1 =  cv2.imread(img_path)
w = image.copy()
w = cv2.cvtColor(draw, cv2.COLOR_BGR2RGB)
 = cv2.FONT_HERSHEY_SIMPLEX
e1 = cv2.putText(image1,lable,(image_row.xmin,image_row.ymin-10),font,1,(0,255,0),2,cv2.LINE_AA)
e1 = cv2.rectangle(image1, (image_row.xmin,image_row.ymin), (image_row.xmax, image_row.ymax), (0,255,0),3)
w_box(draw, box, color=(255, 255, 0))
figure(figsize=(15,15))
axis('off')
subplot(121)
title('original image')
imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
subplot(122)
title('With label and bounding box')
imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))
show()

ay 20 images and with their mask
in range(0,20):
_image_objects(labels_df.iloc[i])
```

```
C:\Users\venky\AppData\Local\Temp\ipykernel_296\6317043.py:16: MatplotlibDeprecationWarning: Auto-removal of overlapping
axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(121)
```



In [22]:

```
# we will remove 'D0w0' class from csv file we will not use that in training model

print(labels_df[labels_df['label'] == 'D0w0'])
labels_df_copy = labels_df.copy()
labels_df_copy = labels_df.drop(labels_df[labels_df['label'] == 'D0w0'].index)
print(labels_df_copy.shape)
```

```
             filename  width  height label  xmin  ymin  xmax  ymax
6971  India_006389.jpg    720     720  D0w0   455   490   570   663
(31879, 8)
```

In [23]:

```
! rm f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/India_006389.jpg
```

```
'rm' is not recognized as an internal or external command,
operable program or batch file.
```

India_006389.jpg this image has only one class 'D0w0' so we will remove from our /content/data/Images dir.

In [24]:

```
# train - test split

X_train,X_test = train_test_split(labels_df_copy, test_size=0.2, random_state=42)

print('shape of total data : ',labels_df_copy.shape)
print('shape of train label  : ',X_train.shape)
print('shape of test label : ',X_test.shape)
```

```
shape of total data :  (31879, 8)
shape of train label  :  (25503, 8)
shape of test label :  (6376, 8)
```

In [25]:

```
sv file for modeling purpose
423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv'
rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv/train_label.csv',index=False)
dd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv/test_label.csv',index = False)
sv('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv/labels_df.csv',index = Fa
```

Now we will Look into label_map.pbtxt file

```
item {
  id: 1
  name: 'D00'
}

item {
  id: 2
  name: 'D10'
}

item {
  id: 3
  name: 'D20'
}

item {
  id: 4
  name: 'D40'
}

```

so label_map.pbtxt file has labels with their id .But they given omly 4 labels. But we have 9 labels so we will edit and modify that file then save in data folder.

item { id: 1 name: 'D00' }

item { id: 2 name: 'D10' }

item { id: 3 name: 'D20' }

item { id: 4 name: 'D40' }

item { id: 5 name: 'D11' }

item { id: 6 name: 'D43' }

item { id: 7 name: 'D44' }

item { id: 8 name: 'D01' }

item { id: 9 name: 'D50' }

In [26]:

```
shutil.copy('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/label_map.pbtxt','f:/rdd08042
```

Out[26]:

```
'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_map.pbtxt'
```

After all that we have new folder 'data' that structure is :

data | |___Annotation | |___Images | |___label_csv | |___label_map.pbtxt

```
data
  |
  |___Annotation
  |
  |___Images
  |
  |___label_csv
  |
  |___label_map.pbtxt
```

Annotation folder have all annotation files

Images folder have all the images

label_csv folder have train and test labels csv files

label_map.pbtxt is modified file.

Observation :

1. We have to total 21041 images and their corresponding xml files
2. We have total 10 classes. In which 'D0w0' is occure only once in Indian dataset. So we can remove that.
3. We have 14569 images that has label.Some of the images have multiple labels.
4. After we convert xml file into csv file the we observe that multiple labeled image has repetaed like

3	Czech_001748.jpg	600	600	D00	174	419	205	455
4	Czech_001748.jpg	600	600	D00	280	451	301	512
5	Czech_001748.jpg	600	600	D00	336	448	383	529
6	Czech_001748.jpg	600	600	D00	295	407	320	444

here image is repeated with different lebels . So we may lead to data leakage problem because these repeated images with different labels might go into test data .So we need to tackle this problem. One thing we can do is split all images randomly into train and test (We will make two folder).Then we will creat xm to csvfile for train and test set.

5. In label_map.pbtxt have map for only 4 labels so we modified that according 9 classes (D0w0 class is removed)

Avoiding Data leakage:

In [27]:

```
image_with_label = list(set(labels_df_copy.filename.values))

print('Number of images have labels : ', len(image_with_label))
```

```
Number of images have labels :  13486
```

In [28]:

```
import random

# creat list of images randomly sample from images that have object
#Since we have ony 14568 images that have labels So we will randomly select from them (80% for train)
train_img = random.sample(image_with_label,k = int(0.8*len(image_with_label)))
```

In [29]:

```
#we creat test set
test_img = list(set(image_with_label) - set(train_img))
```

In [30]:

```
print('Total images for train : ', len(train_img))
print('Total images for test : ',len(test_img))
```

```
Total images for train :  10788
Total images for test :  2698
```

Create seperate folder for train and test images in data dir

In [31]:

```
#! mkdir '/content/data/train_images'

for fil in train_img:
    shutil.copy( 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/'+ fil, "f:/r

file_list = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/c
print('Total train images in train_images folder  : ',len(file_list))
```

```
Total train images in train_images folder  :  10788
```

In [32]:

```
'/content/data/test_images'

n test_img:
l.copy( 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/'+ fil, "f:/rdd080423/

 = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/da
tal train images in train_images folder  : ',len(file_list))
```

```
Total train images in train_images folder  :  2698
```

In [33]:

```
#! mkdir '/content/data/train_annotations'

for fil in train_img:
    shutil.copy( 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Annotations/'+ fil.

file_list = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/
print('Total train images in train_images folder  : ',len(file_list))
```

```
Total train images in train_images folder  :  10788
```

In [34]:

```
#! mkdir '/content/data/test_annotations'

for fil in test_img:
    shutil.copy( 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Annotations/'+ fil.

file_list = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/
print('Total train images in test_annotations folder  : ',len(file_list))
```

```
Total train images in test_annotations folder  :  2698
```

Convert train and test annotation files xml to csv

In [35]:

```
# xml to csv for train set
base_path =  'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/train_images'
train_label_df = xml_to_csv(base_path)

train_label_df.head(10)
```

Out[35]:

	filename	width	height	label	xmin	ymin	xmax	ymax
0	Czech_000047.jpg	600	600	D00	52	415	181	510
1	Czech_000054.jpg	600	600	D00	29	370	235	517
2	Czech_000055.jpg	600	600	D00	147	408	201	448
3	Czech_000055.jpg	600	600	D10	329	489	460	519
4	Czech_000061.jpg	600	600	D00	249	466	269	518
5	Czech_000062.jpg	600	600	D00	303	372	505	518
6	Czech_000062.jpg	600	600	D00	1	400	203	526
7	Czech_000064.jpg	600	600	D40	81	409	134	425
8	Czech_000067.jpg	600	600	D00	366	458	417	542
9	Czech_000081.jpg	600	600	D10	173	489	275	511

In [36]:

```
#xml to csv for test set
base_path =  'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/test_images'
test_label_df = xml_to_csv(base_path)

test_label_df.head(10)
```

Out[36]:

	filename	width	height	label	xmin	ymin	xmax	ymax
0	Czech_000041.jpg	600	600	D20	359	489	479	527
1	Czech_000070.jpg	600	600	D10	7	491	115	507
2	Czech_000070.jpg	600	600	D00	153	425	222	504
3	Czech_000070.jpg	600	600	D00	283	437	352	516
4	Czech_000086.jpg	600	600	D40	215	417	275	449
5	Czech_000086.jpg	600	600	D20	334	422	429	523
6	Czech_000100.jpg	600	600	D00	232	394	289	510
7	Czech_000100.jpg	600	600	D10	81	477	326	522
8	Czech_000100.jpg	600	600	D20	335	449	412	501
9	Czech_000170.jpg	600	600	D00	135	428	175	466

In [37]:

```
#cross checking
print('Total labels in train + test : ',train_label_df.shape[0]+test_label_df.shape[0])
```

```
Total labels in train + test :  31879
```

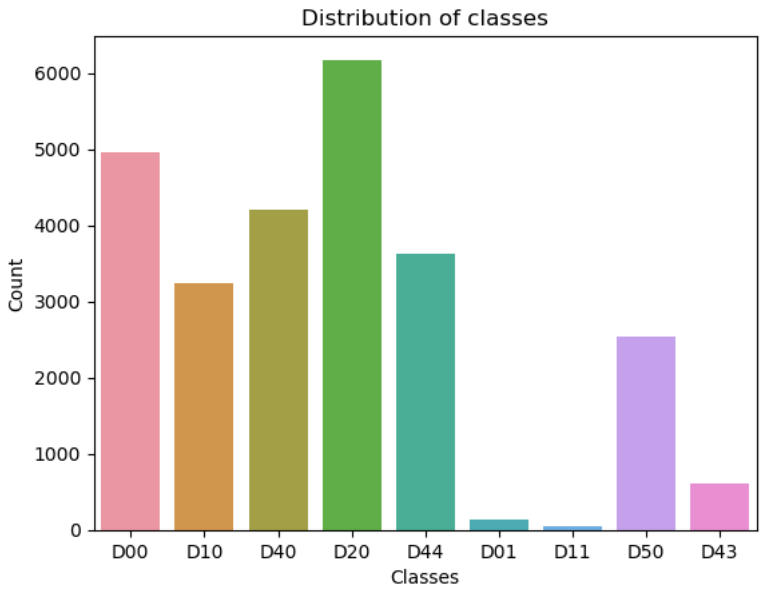
Train Classes Distribution:

In [38]:

```
count_dict = collections.Counter(train_label_df.label)
classes = list(count_dict.keys())
count = []
for label in classes:
  count.append(count_dict[label])

sns.set_palette("winter", 8)
sns.barplot(x=classes,y=count)
plt.xlabel('Classes')
plt.ylabel('Count')
plt.title('Distribution of classes')
```

Out[38]:

```
Text(0.5, 1.0, 'Distribution of classes')
```



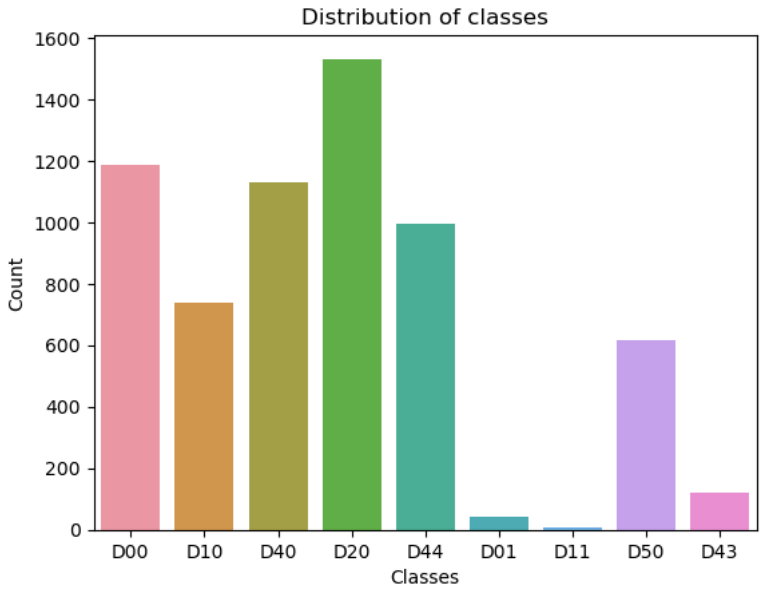
Test Classes Distribution:

In [39]:

```
count_dict = collections.Counter(test_label_df.label)

count = []
for label in classes:
  count.append(count_dict[label])

sns.set_palette("winter", 8)
sns.barplot(x=classes,y=count)
plt.xlabel('Classes')
plt.ylabel('Count')
plt.title('Distribution of classes')
```

Out[39]:

```
Text(0.5, 1.0, 'Distribution of classes')
```



Both Distribution are almost same. So we can say we sampled data in startify manner.

In [40]:

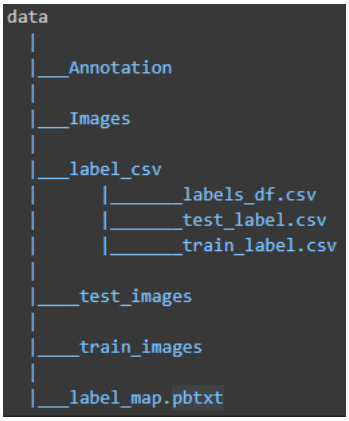
```
test_label_df.to_csv('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv/test_l
train_label_df.to_csv('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv/trair
```

In [41]:

```
!rm -rf /content/data/Images
!rm -rf /content/data/Annotations


!mkdir /content/data/Images
!mkdir /content/data/Annotations


!cp -r '/content/data/train_images' "/content/data/Images"
!cp -r '/content/data/test_images' "/content/data/Images"

!cp -r '/content/data/train_annotations' '/content/data/Annotations'
!cp -r '/content/data/test_annotations' '/content/data/Annotations'

!rm -rf /content/data/train_images
!rm -rf /content/data/test_images

!rm -rf /content/data/train_annotations
!rm -rf /content/data/test_annotations
```

```
'rm' is not recognized as an internal or external command,
operable program or batch file.
'rm' is not recognized as an internal or external command,
operable program or batch file.
The syntax of the command is incorrect.
The syntax of the command is incorrect.
'cp' is not recognized as an internal or external command,
operable program or batch file.
'cp' is not recognized as an internal or external command,
operable program or batch file.
'cp' is not recognized as an internal or external command,
operable program or batch file.
'cp' is not recognized as an internal or external command,
operable program or batch file.
'rm' is not recognized as an internal or external command,
operable program or batch file.
'rm' is not recognized as an internal or external command,
operable program or batch file.
'rm' is not recognized as an internal or external command,
operable program or batch file.
'rm' is not recognized as an internal or external command,
operable program or batch file.
```

My final data folder looks like :



In [42]:

```
#!zip -r /content/data.zip /content/data
```

In [ ]:

In [43]:

```
'''
!mkdir '/content/data/yolo_files'

def xml_to_yolo_train(xml_file_path,image_path,dest_file,class_name_path):
  unique_class_names = []
  file_list = [filename for filename in os.listdir(xml_file_path)]
  with open(dest_file ,'w') as file:
    for xml_file in file_list:
      infile_xml = open(xml_file_path+'/'+xml_file )
      tree = ElementTree.parse(infile_xml)
      root = tree.getroot()
      image_name = str(root.find('filename').text)
      img_path = image_path +'/'+ image_name
      for obj in root.findall('object'):
        class_name = str(obj.find('name').text)
        #print(class_name)
        if class_name not in  unique_class_names:
           unique_class_names.append(class_name)
        cls_id = unique_class_names.index(class_name)
        x_min = str(int(obj.find('bndbox').find('xmin').text))
        y_min = str(int(obj.find('bndbox').find('ymin').text))
        x_max = str(int(obj.find('bndbox').find('xmax').text))
        y_max = str(int(obj.find('bndbox').find('ymax').text))

        obj = x_min+','+y_min+','+x_max+','+y_max+','+str(cls_id)
        img_path += ' '+ obj
      #print(img_path)
      file.write(img_path+'\n')
      #print(unique_class_names)
  with open(class_name_path,'w') as file:
    for name in unique_class_names:
      file.write(str(name)+'\n')
  return unique_class_names

class_name_path = '/content/data/yolo_files/class_names.txt'
dest_file  = '/content/data/yolo_files/train_yolo.txt'
xml_file_path = '/content/data/train_annotations'
image_path = '/content/data/train_images'
classes = xml_to_yolo_train(xml_file_path,image_path,dest_file,class_name_path)
print('classes : ',classes)
'''
```

Out[43]:

```
"\n!mkdir '/content/data/yolo_files'\n\ndef xml_to_yolo_train(xml_file_path,image_path,dest_file,class_name_path):\n  uniqu
e_class_names = []\n  file_list = [filename for filename in os.listdir(xml_file_path)]\n  with open(dest_file ,'w') as fil
e:\n    for xml_file in file_list:\n      infile_xml = open(xml_file_path+'/'+xml_file )\n      tree = ElementTree.parse(in
file_xml)\n      root = tree.getroot()\n      image_name = str(root.find('filename').text)\n      img_path = image_path
+'/'+ image_name\n      for obj in root.findall('object'):\n        class_name = str(obj.find('name').text)\n        #print
(class_name)\n        if class_name not in  unique_class_names:\n           unique_class_names.append(class_name)\n
cls_id = unique_class_names.index(class_name)\n        x_min = str(int(obj.find('bndbox').find('xmin').text))\n        y_mi
n = str(int(obj.find('bndbox').find('ymin').text))\n        x_max = str(int(obj.find('bndbox').find('xmax').text))\n
y_max = str(int(obj.find('bndbox').find('ymax').text))\n\n        obj = x_min+','+y_min+','+x_max+','+y_max+','+str(cls_id)
\n        img_path += ' '+ obj\n      #print(img_path)\n      file.write(img_path+'\n')\n      #print(unique_class_names)\n
with open(class_name_path,'w') as file:\n    for name in unique_class_names:\n      file.write(str(name)+'\n')\n  return un
ique_class_names\n\nclass_name_path = '/content/data/yolo_files/class_names.txt'\ndest_file  = '/content/data/yolo_files/tr
ain_yolo.txt'\nxml_file_path = '/content/data/train_annotations'\nimage_path = '/content/data/train_images'\nclasses = xml_
to_yolo_train(xml_file_path,image_path,dest_file,class_name_path)\nprint('classes : ',classes)\n"
```

In [44]:

```
'''
def xml_to_yolo_test(xml_file_path,image_path,dest_file,classes):
  unique_class_names = classes
  file_list = [filename for filename in os.listdir(xml_file_path)]
  with open(dest_file ,'w') as file:
    for xml_file in file_list:
      infile_xml = open(xml_file_path+'/'+xml_file )
      tree = ElementTree.parse(infile_xml)
      root = tree.getroot()
      image_name = str(root.find('filename').text)
      img_path = image_path +'/'+ image_name
      for obj in root.findall('object'):
        class_name = str(obj.find('name').text)
        #print(class_name)
        if class_name not in  unique_class_names:
           unique_class_names.append(class_name)
        cls_id = unique_class_names.index(class_name)
        x_min = str(int(obj.find('bndbox').find('xmin').text))
        y_min = str(int(obj.find('bndbox').find('ymin').text))
        x_max = str(int(obj.find('bndbox').find('xmax').text))
        y_max = str(int(obj.find('bndbox').find('ymax').text))

        obj = x_min+','+y_min+','+x_max+','+y_max+','+str(cls_id)
        img_path += ' '+ obj
      #print(img_path)
      file.write(img_path+'\n')
      #print(unique_class_names)
  return unique_class_names

dest_file  = '/content/data/yolo_files/test_yolo.txt'
xml_file_path = '/content/data/test_annotations'
image_path = '/content/data/test_images'
classes = xml_to_yolo_test(xml_file_path,image_path,dest_file,classes)
print(classes)
'''
```

Out[44]:

```
"\ndef xml_to_yolo_test(xml_file_path,image_path,dest_file,classes):\n  unique_class_names = classes\n  file_list = [filena
me for filename in os.listdir(xml_file_path)]\n  with open(dest_file ,'w') as file:\n    for xml_file in file_list:\n
infile_xml = open(xml_file_path+'/'+xml_file )\n      tree = ElementTree.parse(infile_xml)\n      root = tree.getroot()\n
image_name = str(root.find('filename').text)\n      img_path = image_path +'/'+ image_name\n      for obj in root.findall
('object'):\n        class_name = str(obj.find('name').text)\n        #print(class_name)\n        if class_name not in  uni
que_class_names:\n           unique_class_names.append(class_name)\n        cls_id = unique_class_names.index(class_name)\n
x_min = str(int(obj.find('bndbox').find('xmin').text))\n        y_min = str(int(obj.find('bndbox').find('ymin').text))\n
x_max = str(int(obj.find('bndbox').find('xmax').text))\n        y_max = str(int(obj.find('bndbox').find('ymax').text))\n\n
obj = x_min+','+y_min+','+x_max+','+y_max+','+str(cls_id)\n        img_path += ' '+ obj\n      #print(img_path)\n      fil
e.write(img_path+'\n')\n      #print(unique_class_names)\n  return unique_class_names\n\ndest_file  = '/content/data/yolo_f
iles/test_yolo.txt'\nxml_file_path = '/content/data/test_annotations'\nimage_path = '/content/data/test_images'\nclasses =
xml_to_yolo_test(xml_file_path,image_path,dest_file,classes)\nprint(classes)\n"
```

In [45]:

```
'''
with open('/content/data/train_annotations/Japan_007529.xml', 'r') as f:
    data = f.read()

Bs_data = BeautifulSoup(data, "xml")
Bs_data'''
```

Out[45]:

```
'\nwith open(\'/content/data/train_annotations/Japan_007529.xml\', \'r\') as f: \n    data = f.read()\n\nBs_data = Beautifu
lSoup(data, "xml") \nBs_data'
```

Type *Markdown* and LaTeX: α^2

Augmentation

1. We will resize and augment train images.
2. We will only resize the test images.
3. Resize of the image according model we are going to use for fine tunning.

Reference :

https://github.com/asetkn/Tutorial-Image-and-Multiple-Bounding-Boxes-Augmentation-for-Deep-Learning-in-4-Steps/blob/master/Tutorial-Image-and-Multiple-Bounding-Boxes-Augmentation-for-Deep-Learning-in-4-Steps.ipynb (https://github.com/asetkn/Tutorial-Image-and-Multiple-Bounding-Boxes-Augmentation-for-Deep-Learning-in-4-Steps/blob/master/Tutorial-Image-and-Multiple-Bounding-Boxes-Augmentation-for-Deep-Learning-in-4-Steps.ipynb)

Resize Images according to model

In [49]:

```
grp = train_label_df.groupby('filename')
grp_df = grp.get_group('Japan_010845.jpg')
grp_df = grp_df.reset_index()
grp_df = grp_df.drop(['index'], axis=1)
grp_df
```

Out[49]:

	filename	width	height	label	xmin	ymin	xmax	ymax
0	Japan_010845.jpg	1024	1024	D40	380	340	487	396
1	Japan_010845.jpg	1024	1024	D40	380	219	495	269
2	Japan_010845.jpg	1024	1024	D40	652	257	901	384
3	Japan_010845.jpg	1024	1024	D40	394	421	530	477

In [50]:

```
# get bounding boxes coordinates from grouped data frame and write into array
bb_array =grp_df.drop(['filename', 'width', 'height', 'label'], axis=1).values
# display the array we've got
bb_array
```

Out[50]:

```
array([[380, 340, 487, 396],
       [380, 219, 495, 269],
       [652, 257, 901, 384],
       [394, 421, 530, 477]], dtype=int64)
```

In [57]:

```
image = imageio.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/Japan_0
bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)

# display the image and draw bounding boxes
print('1024 x 1024 image: ')

import matplotlib.pyplot as plt

w, h = image.shape[1]/100, image.shape[0]/100
dpi = 100

fig, ax = plt.subplots(figsize=(w, h), dpi=dpi)
ax.set_title("imgaug.imshow(%s)" % (image.shape,))
ax.imshow(bbs.draw_on_image(image, size=4))
plt.show()
```

```
C:\Users\venky\AppData\Local\Temp\ipykernel_296\2842839634.py:1: DeprecationWarning: Starting with ImageIO v3 the behavior
of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear) use `
import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/da
ta/Images/Japan_010894.jpg')
```

```
1024 x 1024 image:
```

Resize train Images

In [58]:

```
# function to convert BoundingBoxesOnImage object into DataFrame
def bbs_obj_to_df(bbs_object):
#     convert BoundingBoxesOnImage object into array
    bbs_array = bbs_object.to_xyxy_array()
   # print(bbs_array)
#     convert array into a DataFrame ['xmin', 'ymin', 'xmax', 'ymax'] columns
    df_bbs = pd.DataFrame(bbs_array, columns=['xmin', 'ymin', 'xmax', 'ymax'])
    return df_bbs
```

In [59]:

```
def resize_img(label_dataframe,height,width, image_path, aug_images_path, image_prefix):
    # create data frame which we're going to populate with augmented image info
    aug_bbs_xy = pd.DataFrame(columns=
                              ['filename','width','height','label', 'xmin', 'ymin', 'xmax', 'ymax']
                             )
    grouped = label_dataframe.groupby('filename')

    for filename in label_dataframe['filename'].unique():
    #   Get separate data frame grouped by file name
        group_df = grouped.get_group(filename)
        group_df = group_df.reset_index()
        group_df = group_df.drop(['index'], axis=1)

    #   The only difference between if and elif statements below is the use of height_resize and width_resize augmentors
    #   deffined previously.
        image = imageio.imread(image_path+filename)
        #   get bounding boxes coordinates and write into array
        bb_array = group_df.drop(['filename', 'width', 'height', 'label'], axis=1).values
        #   pass the array of bounding boxes coordinates to the imgaug library
        bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)
        #   apply augmentation on image and on the bounding boxes
        image_aug, bbs_aug = resize(image=image, bounding_boxes=bbs)
        #   write augmented image to a file
        imageio.imwrite(aug_images_path+image_prefix+filename, image_aug)
        #   create a data frame with augmented values of image width and height
        info_df = group_df.drop(['xmin', 'ymin', 'xmax', 'ymax'], axis=1)
        for index, _ in info_df.iterrows():
            info_df.at[index, 'width'] = image_aug.shape[1]
            info_df.at[index, 'height'] = image_aug.shape[0]
        #   rename filenames by adding the predifined prefix
        info_df['filename'] = info_df['filename'].apply(lambda x: image_prefix+x)
        #   create a data frame with augmented bounding boxes coordinates using the function we created earlier
        bbs_df = bbs_obj_to_df(bbs_aug)
        #   concat all new augmented info into new data frame
        aug_df = pd.concat([info_df, bbs_df], axis=1)
        #   append rows to aug_bbs_xy data frame
        aug_bbs_xy = pd.concat([aug_bbs_xy, aug_df])

    # return dataframe with updated images and bounding boxes annotations
    aug_bbs_xy = aug_bbs_xy.reset_index()
    aug_bbs_xy = aug_bbs_xy.drop(['index'], axis=1)
    return aug_bbs_xy
```

In [60]:

```

# to resize the images we create two augmenters
# one is used when the image height is more than 600px and the other when the width is more than 600px

height = 640
width = 640

resize = iaa.Sequential([
    iaa.Resize({"height": height, "width": width})
])



# apply resizing augmentation to our images and write the updated images and bounding boxes annotations to the DataFrame
# we will not apply prefix to our files and will overwrite images in the same directory
image_path='f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/train_images/'
aug_images_path = 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/train_images

resized_train_images_df = resize_img(label_dataframe=train_label_df,height = height,width=width,
                                  image_path = image_path,aug_images_path=aug_images_path, image_prefix = '')
```

```
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\4047177531.py:16: DeprecationWarning: Starting with ImageIO v3 the behav
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\4047177531.py:16: DeprecationWarning: Starting with ImageIO v3 the behav
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\4047177531.py:16: DeprecationWarning: Starting with ImageIO v3 the behav
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\4047177531.py:16: DeprecationWarning: Starting with ImageIO v3 the behav
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
```

In [61]:

```
resized_train_images_df.head(10)
```

Out[61]:

	filename	width	height	label	xmin	ymin	xmax	ymax
0	Czech_000047.jpg	640	640	D00	55.466667	442.666656	193.066681	544.000000
1	Czech_000054.jpg	640	640	D00	30.933332	394.666687	250.666672	551.466675
2	Czech_000055.jpg	640	640	D00	156.800003	435.200012	214.400009	477.866669
3	Czech_000055.jpg	640	640	D10	350.933350	521.599976	490.666656	553.599976
4	Czech_000061.jpg	640	640	D00	265.600006	497.066650	286.933319	552.533325
5	Czech_000062.jpg	640	640	D00	323.200012	396.799988	538.666626	552.533325
6	Czech_000062.jpg	640	640	D00	1.066667	426.666687	216.533340	561.066650
7	Czech_000064.jpg	640	640	D40	86.400002	436.266663	142.933334	453.333313
8	Czech_000067.jpg	640	640	D00	390.400024	488.533325	444.799988	578.133301
9	Czech_000081.jpg	640	640	D10	184.533325	521.599976	293.333344	545.066650

In [62]:

```
resized_train_images_df.isnull().sum()
```

Out[62]:

```
filename    0
width       0
height      0
label       0
xmin        0
ymin        0
xmax        0
ymax        0
dtype: int64
```

Resize test Images

In [63]:

```
# to resize the images we create two augmenters
# one is used when the image height is more than 600px and the other when the width is more than 600px

height = 640
width = 640

resize = iaa.Sequential([
    iaa.Resize({"height": height, "width": width})
])



# apply resizing augmentation to our images and write the updated images and bounding boxes annotations to the DataFrame
# we will not apply prefix to our files and will overwrite images in the same directory
image_path='f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/test_images/'
aug_images_path = 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/test_images/
#label_dataframe = test_label_df
resized_test_images_df = resize_img(label_dataframe=test_label_df,height = height,width=width,
                                    image_path = image_path,aug_images_path=aug_images_path, image_prefix = '')
```

```
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\4047177531.py:16: DeprecationWarning: Starting with ImageIO v3 the behav
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\4047177531.py:16: DeprecationWarning: Starting with ImageIO v3 the behav
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\4047177531.py:16: DeprecationWarning: Starting with ImageIO v3 the behav
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\4047177531.py:16: DeprecationWarning: Starting with ImageIO v3 the behav
ior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path+filename)
```

In [64]:

```
resized_test_images_df.head(10)
```

Out[64]:

	filename	width	height	label	xmin	ymin	xmax	ymax
0	Czech_000041.jpg	640	640	D20	382.933350	521.599976	510.933350	562.133301
1	Czech_000070.jpg	640	640	D10	7.466667	523.733337	122.666664	540.800049
2	Czech_000070.jpg	640	640	D00	163.199997	453.333313	236.800003	537.599976
3	Czech_000070.jpg	640	640	D00	301.866669	466.133362	375.466644	550.400024
4	Czech_000086.jpg	640	640	D40	229.333328	444.799988	293.333344	478.933350
5	Czech_000086.jpg	640	640	D20	356.266663	450.133331	457.599976	557.866699
6	Czech_000100.jpg	640	640	D00	247.466660	420.266693	308.266663	544.000000
7	Czech_000100.jpg	640	640	D10	86.400002	508.800018	347.733337	556.799988
8	Czech_000100.jpg	640	640	D20	357.333344	478.933350	439.466675	534.399963
9	Czech_000170.jpg	640	640	D00	144.000000	456.533325	186.666656	497.066650

In [65]:

```
resized_test_images_df.isnull().sum()
```

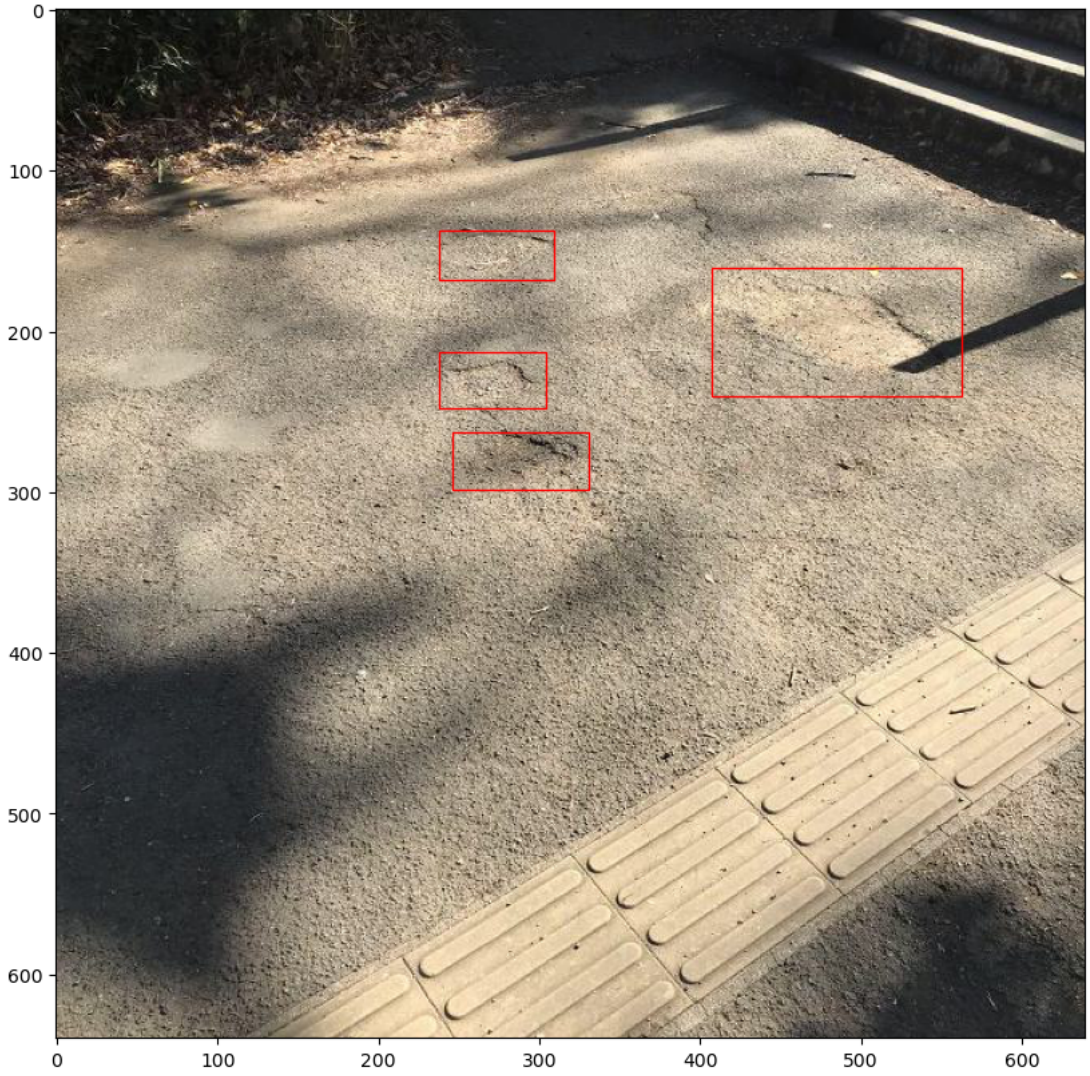
Out[65]:

```
filename    0
width       0
height      0
label       0
xmin        0
ymin        0
xmax        0
ymax        0
dtype: int64
```

In [72]:

```
import matplotlib.pyplot as plt

# visualise the resized 'Japan_010845.jpg' image with bounding boxes
# to make sure our bounding boxes were resized correctly as well
grouped = resized_train_images_df.groupby('filename')
group_df = grouped.get_group('Japan_010845.jpg')
group_df = group_df.reset_index()
group_df = group_df.drop(['index'], axis=1)
bb_array = group_df.drop(['filename', 'width', 'height', 'label'], axis=1).values
image = imageio.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/train_i
bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)

# plot the image with bounding boxes
fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(image)

for i in range(len(bb_array)):
    xmin, ymin, xmax, ymax = bb_array[i]
    rect = plt.Rectangle((xmin, ymin), xmax - xmin, ymax - ymin, fill=False, color='r')
    ax.add_patch(rect)

plt.show()
```

```
C:\Users\venky\AppData\Local\Temp\ipykernel_296\1385662491.py:10: DeprecationWarning: Starting with ImageIO v3 the behavior
of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear) use `
import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/da
ta/Images/train_images/Japan_010845.jpg')
```



Visualising Resized Test Images:

In [73]:

```
#grouped_original = test_label_df.groupby('filename')
grouped_resized = resized_test_images_df.groupby('filename')

for filename in resized_test_images_df['filename'].unique()[0:20]:
    fig, cell = plt.subplots(1, 1, figsize=(7,7))

    group_r_df = grouped_resized.get_group(filename)
    group_r_df = group_r_df.reset_index()
    group_r_df = group_r_df.drop(['index'], axis=1)
    bb_r_array = group_r_df.drop(['filename', 'width', 'height'], axis=1).values
    resized_img = imageio.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Imag
    img_array1 = np.asarray(resized_img)
    cell.imshow(img_array1)
    #bbs_r = BoundingBoxesOnImage.from_xyxy_array(bb_r_array, shape=resized_img.shape)
    #print(bb_r_array)
    for box in bb_r_array:
      xmin = box[1]
      ymin = box[2]
      xmax = box[3]
      ymax = box[4]
     # color = colors[int(box[0])]
      label = '{}'.format(box[0])
      cell.set_title('Resized image 640x640',fontsize = 18)
      cell.add_patch(plt.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin, fill=False, linewidth=2))
      cell.text(xmin, ymin, label, size='large', color='black', bbox={'alpha':0.5})
```

Augmenting Train Images:

In [ ]:

```
#!rm -rf  '/content/data/Images/augmented_image'
```

In [ ]:

```
#!mkdir  '/content/data/Images/augmented_image'
```

In [74]:

```
def image_aug(df, images_path, aug_images_path, image_prefix, augmentor):
    # create data frame which we're going to populate with augmented image info
    aug_bbs_xy = pd.DataFrame(columns=
                              ['filename','width','height','label', 'xmin', 'ymin', 'xmax', 'ymax']
                             )
    grouped = df.groupby('filename')

    for filename in df['filename'].unique():
    #   get separate data frame grouped by file name
        group_df = grouped.get_group(filename)
        group_df = group_df.reset_index()
        group_df = group_df.drop(['index'], axis=1)
    #   read the image
        image = imageio.imread(images_path+filename)
    #   get bounding boxes coordinates and write into array
        bb_array = group_df.drop(['filename', 'width', 'height', 'label'], axis=1).values
    #   pass the array of bounding boxes coordinates to the imgaug library
        bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)
    #   apply augmentation on image and on the bounding boxes
        image_aug, bbs_aug = augmentor(image=image, bounding_boxes=bbs)
    #   disregard bounding boxes which have fallen out of image pane
        bbs_aug = bbs_aug.remove_out_of_image()
    #   clip bounding boxes which are partially outside of image pane
        bbs_aug = bbs_aug.clip_out_of_image()

    #   don't perform any actions with the image if there are no bounding boxes left in it
        if re.findall('Image...', str(bbs_aug)) == ['Image([]']:
            pass

    #   otherwise continue
        else:
        #   write augmented image to a file
            imageio.imwrite(aug_images_path+image_prefix+filename, image_aug)
        #   create a data frame with augmented values of image width and height
            info_df = group_df.drop(['xmin', 'ymin', 'xmax', 'ymax'], axis=1)
            for index, _ in info_df.iterrows():
                info_df.at[index, 'width'] = image_aug.shape[1]
                info_df.at[index, 'height'] = image_aug.shape[0]
        #   rename filenames by adding the predifined prefix
            info_df['filename'] = info_df['filename'].apply(lambda x: image_prefix+x)
        #   create a data frame with augmented bounding boxes coordinates using the function we created earlier
            bbs_df = bbs_obj_to_df(bbs_aug)
        #   concat all new augmented info into new data frame
            aug_df = pd.concat([info_df, bbs_df], axis=1)
        #   append rows to aug_bbs_xy data frame
            aug_bbs_xy = pd.concat([aug_bbs_xy, aug_df])

    # return dataframe with updated images and bounding boxes annotations
    aug_bbs_xy = aug_bbs_xy.reset_index()
    aug_bbs_xy = aug_bbs_xy.drop(['index'], axis=1)
    return aug_bbs_xy
```

In [75]:

```
# This setup of augmentation parameters will pick 3 of 8 given augmenters and apply them in random order
aug = iaa.SomeOf(3, [
    iaa.Affine(scale=(0.6,1.2)),#zoom in or zoom out
    iaa.Affine(rotate=(-8, 8)),#rotation
    #iaa.Affine(translate_percent={"x": (-0.3, 0.3), "y": (-0.3, 0.3)}),
    iaa.LinearContrast((0.5,1.5),per_channel=True),#contrast
    iaa.Fliplr(0.4),#mirror image horizontally
    #iaa.Flipud(0.1),#mirror image vartically
    iaa.Multiply((0.5, 1.5)),#multipy pixels
    iaa.GaussianBlur(sigma=(1.0, 3.0)),#blur image
    iaa.AdditiveGaussianNoise(scale=(0.03*255, 0.05*255)),#adding gausian noise
    iaa.Sharpen(alpha=(0.5), lightness=(0.75,1.5)) #sharpen images
])


# Apply augmentation to our images and save files into 'augmented_image/' folder with 'aug_' prefix.
# Write the updated images and bounding boxes annotations to the augmented_images_df dataframe.
input_image_path = 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/train_image
aug_image_path = 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/augmented_ima
aug_prefix = 'aug_'   #prefix to be add to identify augmented image (optional)
augmentor = aug   #callinf aug fuction which is define above
augmented_train_images_df = image_aug(resized_train_images_df,input_image_path,aug_image_path, aug_prefix, augmentor)
```

```
or of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear)
use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(images_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\742704043.py:14: DeprecationWarning: Starting with ImageIO v3 the behavi
or of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear)
use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(images_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\742704043.py:14: DeprecationWarning: Starting with ImageIO v3 the behavi
or of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear)
use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(images_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\742704043.py:14: DeprecationWarning: Starting with ImageIO v3 the behavi
or of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear)
use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(images_path+filename)
C:\Users\venky\AppData\Local\Temp\ipykernel_296\742704043.py:14: DeprecationWarning: Starting with ImageIO v3 the behavi
or of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear)
use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(images_path+filename)
```

In [76]:

```
augmented_train_images_df
```

Out[76]:

	filename	width	height	label	xmin	ymin	xmax	ymax
0	aug_Czech_000047.jpg	640	640	D00	8.829964	464.292542	170.688583	583.490784
1	aug_Czech_000054.jpg	640	640	D00	32.757946	396.267212	256.096985	558.161377
2	aug_Czech_000055.jpg	640	640	D00	199.093750	405.345612	241.766541	436.955078
3	aug_Czech_000055.jpg	640	640	D10	342.916870	469.354767	446.437897	493.061890
4	aug_Czech_000061.jpg	640	640	D00	279.522552	451.750122	295.396057	493.021240
...	...	...	...	...	...	...	...	...
25507	aug_Japan_013129.jpg	640	640	D00	208.000000	475.733337	240.000000	523.733337
25508	aug_Japan_013131.jpg	640	640	D20	506.666687	538.666626	602.666687	579.199951
25509	aug_Japan_013132.jpg	640	640	D00	576.997864	281.758514	640.000000	349.023651
25510	aug_Japan_013132.jpg	640	640	D43	193.253967	83.368774	430.107727	127.085350
25511	aug_Japan_013132.jpg	640	640	D20	254.857483	168.307632	362.779877	443.684082

25512 rows × 8 columns

In [77]:

```
augmented_train_images_df.isnull().sum()
```

Out[77]:

```
filename     0
width        0
height       0
label        0
xmin        28
ymin        28
xmax        28
ymax        28
dtype: int64
```

Due to augmentation some boxes might be out of image plane so that boxes will shows ans NaN .So we will remove Nan's In from datafram.

In [78]:

```
augmented_train_images_df=augmented_train_images_df.dropna()
augmented_train_images_df.isnull().sum()
```

Out[78]:

```
filename    0
width       0
height      0
label       0
xmin        0
ymin        0
xmax        0
ymax        0
dtype: int64
```

In [79]:

```
'''
grouped_resized = resized_images_df.groupby('filename')
grouped_augmented = augmented_images_df.groupby('filename')

for filename in resized_images_df['filename'].unique()[0:20]:

    group_r_df = grouped_resized.get_group(filename)
    group_r_df = group_r_df.reset_index()
    group_r_df = group_r_df.drop(['index'], axis=1)
    bb_r_array = group_r_df.drop(['filename', 'width', 'height', 'label'], axis=1).values
    resized_img = imageio.imread('/content/data/Images/train_images/'+filename)
    bbs_r = BoundingBoxesOnImage.from_xyxy_array(bb_r_array, shape=resized_img.shape)

    print(filename)
    group_a_df = grouped_augmented.get_group('aug_'+filename)
    group_a_df = group_a_df.reset_index()
    group_a_df = group_a_df.drop(['index'], axis=1)
    bb_a_array = group_a_df.drop(['filename', 'width', 'height', 'label'], axis=1).values
    augmented_img = imageio.imread('/content/data/Images/augmented_image/'+'aug_'+filename)
    bbs_a = BoundingBoxesOnImage.from_xyxy_array(bb_a_array, shape=augmented_img.shape)

    ia.imshow(np.hstack([
            bbs_r.draw_on_image(resized_img, size=2),
            bbs_a.draw_on_image(augmented_img, size=2)
            ]))
'''
```

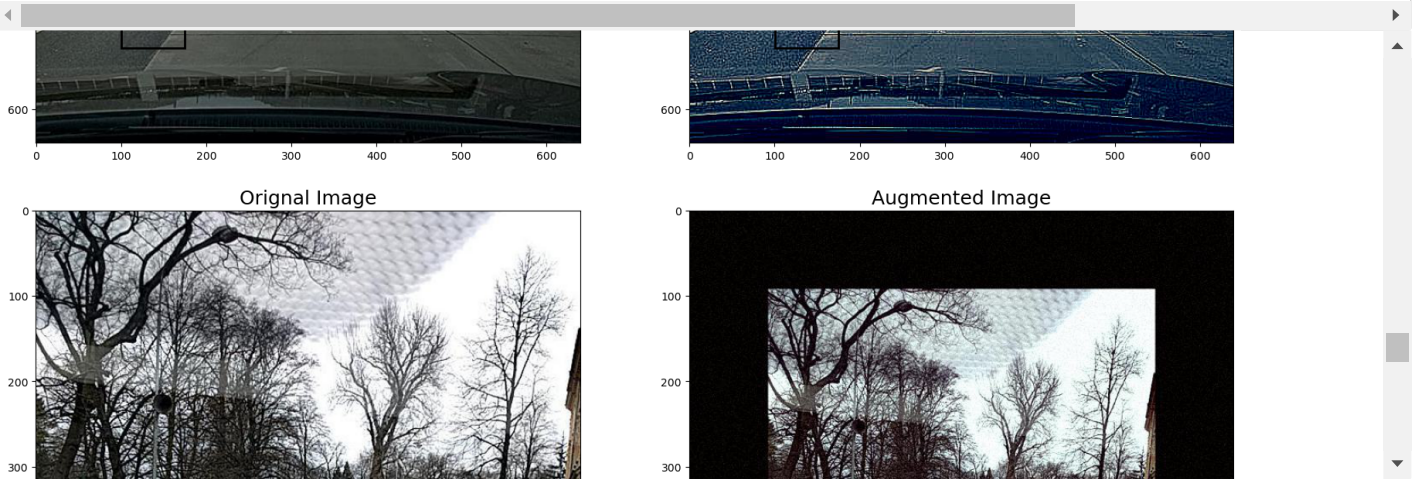
Out[79]:

```
"\ngrouped_resized = resized_images_df.groupby('filename')\ngrouped_augmented = augmented_images_df.groupby('filename')\n\n
for filename in resized_images_df['filename'].unique()[0:20]:    \n    \n    group_r_df = grouped_resized.get_group(filenam
e)\n    group_r_df = group_r_df.reset_index()\n    group_r_df = group_r_df.drop(['index'], axis=1)\n    bb_r_array = group_
r_df.drop(['filename', 'width', 'height', 'label'], axis=1).values\n    resized_img = imageio.imread('/content/data/Images/
train_images/'+filename)\n    bbs_r = BoundingBoxesOnImage.from_xyxy_array(bb_r_array, shape=resized_img.shape)\n    \n
print(filename)\n    group_a_df = grouped_augmented.get_group('aug_'+filename)\n    group_a_df = group_a_df.reset_index()\n
group_a_df = group_a_df.drop(['index'], axis=1)\n    bb_a_array = group_a_df.drop(['filename', 'width', 'height', 'label'],
axis=1).values\n    augmented_img = imageio.imread('/content/data/Images/augmented_image/'+'aug_'+filename)\n    bbs_a = Bo
undingBoxesOnImage.from_xyxy_array(bb_a_array, shape=augmented_img.shape)\n    \n    ia.imshow(np.hstack([\n            bbs
_r.draw_on_image(resized_img, size=2),\n            bbs_a.draw_on_image(augmented_img, size=2)\n            ]))\n"
```

Visualising Augmented vs. Resized Train Images

In [80]:

```
grouped_resized = resized_train_images_df.groupby('filename')
grouped_augmented = augmented_train_images_df.groupby('filename')

for filename in resized_train_images_df['filename'].unique()[0:20]:
    fig, cell = plt.subplots(1, 2, figsize=(20,16))

    group_r_df = grouped_resized.get_group(filename)
    group_r_df = group_r_df.reset_index()
    group_r_df = group_r_df.drop(['index'], axis=1)
    bb_r_array = group_r_df.drop(['filename', 'width', 'height'], axis=1).values
    resized_img = imageio.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Imag
    img_array1 = np.asarray(resized_img)
    cell[0].imshow(img_array1)
    #bbs_r = BoundingBoxesOnImage.from_xyxy_array(bb_r_array, shape=resized_img.shape)
    #print(bb_r_array)
    for box in bb_r_array:
      xmin = box[1]
      ymin = box[2]
      xmax = box[3]
      ymax = box[4]
     # color = colors[int(box[0])]
      label = '{}'.format(box[0])
      cell[0].set_title('Orignal Image',fontsize = 18)
      cell[0].add_patch(plt.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin, fill=False, linewidth=2))
      cell[0].text(xmin, ymin, label, size='x-large', color='black', bbox={'alpha':0.5})



    group_a_df = grouped_augmented.get_group('aug_'+filename)
    group_a_df = group_a_df.reset_index()
    group_a_df = group_a_df.drop(['index'], axis=1)
    bb_a_array = group_a_df.drop(['filename', 'width', 'height',], axis=1).values
    augmented_img = imageio.imread('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Im
    img_array2 = np.asarray(augmented_img)
    cell[1].imshow(img_array2)
   # bbs_a = BoundingBoxesOnImage.from_xyxy_array(bb_a_array, shape=augmented_img.shape)
    for box in   bb_a_array :
      xmin = box[1]
      ymin = box[2]
      xmax = box[3]
      ymax = box[4]
      #color = colors[int(box[0])]
      label = '{}'.format(box[0])
      cell[1].set_title('Augmented Image',fontsize = 18)
      cell[1].add_patch(plt.Rectangle((xmin, ymin), xmax-xmin, ymax-ymin, fill=False, linewidth=2))
      cell[1].text(xmin, ymin, label, size='x-large', color='black', bbox={'alpha':0.5})
```



In [81]:

```
augmented_train_images_df.filename.values
```

Out[81]:

```
array(['aug_Czech_000047.jpg', 'aug_Czech_000054.jpg',
       'aug_Czech_000055.jpg', ..., 'aug_Japan_013132.jpg',
       'aug_Japan_013132.jpg', 'aug_Japan_013132.jpg'], dtype=object)
```

In [82]:

```
for fil in augmented_train_images_df.filename.values:
    shutil.copy( 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/augmented_ima
```

In [ ]:

```
!rm -rf /content/data/Images/augmented_image
```

In [83]:

```
file_list = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/c
print('Total train images in train_images folder  : ',len(file_list))
```

```
Total train images in train_images folder  :  21575
```

In [84]:

```
#combining resized train images dataframe and augmented dataframes together.
org_aug_train_combine = pd.concat([resized_train_images_df,augmented_train_images_df])
org_aug_train_combine.shape
```

Out[84]:

```
(50997, 8)
```

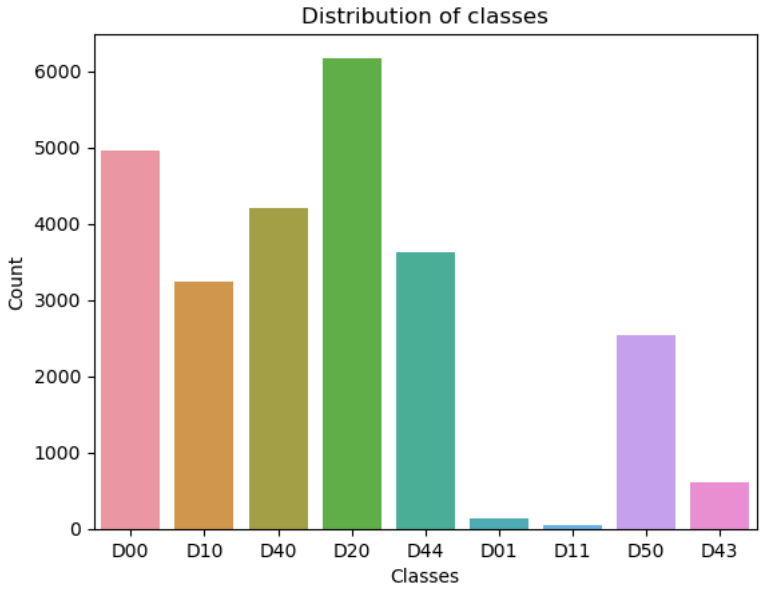
In [86]:

```
count_dict = collections.Counter(train_label_df.label)
#classes = list(count_dict.keys())
print(count_dict)
count = []
for label in classes:
  count.append(count_dict[label])

sns.set_palette("winter", 8)
sns.barplot(x=classes,y=count)
plt.xlabel('Classes')
plt.ylabel('Count')
plt.title('Distribution of classes')
```

```
Counter({'D20': 6175, 'D00': 4951, 'D40': 4214, 'D44': 3623, 'D10': 3243, 'D50': 2532, 'D43': 601, 'D01': 135, 'D11': 39})
```

Out[86]:

```
Text(0.5, 1.0, 'Distribution of classes')
```



In [87]:

```
org_aug_train_combine.isnull().sum()
```

Out[87]:

```
filename    0
width       0
height      0
label       0
xmin        0
ymin        0
xmax        0
ymax        0
dtype: int64
```

In [88]:

```
vert dataframe to csv files
zed_train_images_df.to_csv('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv/
zed_test_images_df.to_csv('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv/t
ented_train_images_df.to_csv('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_cs
aug_train_combine.to_csv('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/label_csv/or
```

In [89]:

```
file_list = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/c
print('Total train images in train_images folder  : ',len(file_list))

file_list = [filename for filename in os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/c
print('Total train images in test_images folder  : ',len(file_list))
```

```
Total train images in train_images folder  :  21575
Total train images in test_images folder  :  2698
```

In [ ]:

```
!zip -r /content/data.zip /content/data
```

In [92]:

```
shutil.copy('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/generate_tfrecord.py' , 'f:/rd
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[92], line 1
----> 1 shutil.copy('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/generat
e_tfrecord.py' , 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data')

File F:\anaconda3\lib\shutil.py:417, in copy(src, dst, follow_symlinks)
    415 if os.path.isdir(dst):
    416     dst = os.path.join(dst, os.path.basename(src))
--> 417 copyfile(src, dst, follow_symlinks=follow_symlinks)
    418 copymode(src, dst, follow_symlinks=follow_symlinks)
    419 return dst

File F:\anaconda3\lib\shutil.py:254, in copyfile(src, dst, follow_symlinks)
    252     os.symlink(os.readlink(src), dst)
    253 else:
--> 254     with open(src, 'rb') as fsrc:
    255         try:
    256             with open(dst, 'wb') as fdst:
    257                 # macOS

FileNotFoundError: [Errno 2] No such file or directory: 'f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Mul
tiple-Countries-main/content/generate_tfrecord.py'
```

In [ ]:

```
!rm -r /content/drive/MyDrive/object_detection/data
```

In [ ]:

```
!cp -avr /content/data /content/drive/MyDrive/object_detection
```

In [91]:

```
os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/train_images')]
ges folder  : ',len(file_list))

os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/test_images')]
es folder  : ',len(file_list))

os.listdir('f:/rdd080423/Transfer-Learning-based-Road-Damage-Detection-for-Multiple-Countries-main/content/data/Images/augmented_image')]
_images folder  : ',len(file_list))
```

```
Total train images in train_images folder  :  21575
Total train images in test_images folder  :  2698
Total train images in augmented_images folder  :  10787
```

Here we zipped folder into drive for training SSD_mobilnet and Faster_RCNN.

In [ ]: