German Traffic Signs Detection

The German Traffic Sign Detection Benchmark is an object detection problem where the task at hand is to detect traffic signs. Traffic sign detection is still a challenging real-world problem of high industrial relevance.Participating algorithms need to pinpoint the location of given categories of traffic signs (prohibitory, mandatory or danger)

The Dataset Information is as follows : A single-image detection problem

● 900 Images (divided in 600 training images and 300 evaluation images)

● Division into four categories that suit the properties of various detection approaches with different properties

In [7]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
```

Data-Set Acquisition

http://benchmark.ini.rub.de/?section=gtsdb&subsection=about (http://benchmark.ini.rub.de/?section=gtsdb&subsection=about)

Dataset Format

Image Format

- The images contain zero to six traffic signs. However, even if there is a traffic sign located in the image it may not belong to the competition relevant categories (prohibitive, danger, mandatory).
- Images are stored in PPM format
- The sizes of the traffic signs in the images vary from 16x16 to 128x128
- Traffic signs may appear in every perspective and under every lighting condition

Anottation Format

- Annotations are provided in CSV files. Fields are seperated by a semicolon (;). They contain the following information:
  - Filename: Filename of the image the annotations apply for
  - Traffic sign's region of interest (ROI) in the image
  - Leftmost image column of the ROI
  - Upmost image row of the ROI
  - Rightmost image column of the ROI
  - Downmost image row of the ROI

- ID providing the traffic sign's class

Task of the Problem

- **Our model should predict whether a given image belongs to any of the four classes namely "Prohibitory" , "Danger", "Mandatory" and "Other"**
- **There are 42 classes of traffic signs out of which they are grouped under 4 classes as mentioned above**

In [4]:

```
signnames=pd.read_csv("E:\\GTSDB\\signnames.csv")
signnames
```

Out[4]:

	ClassId	SignName
0	0	Speed limit (20km/h)
1	1	Speed limit (30km/h)
2	2	Speed limit (50km/h)
3	3	Speed limit (60km/h)
4	4	Speed limit (70km/h)
5	5	Speed limit (80km/h)
6	6	End of speed limit (80km/h)
7	7	Speed limit (100km/h)
8	8	Speed limit (120km/h)
9	9	No passing

EDA And Loading the Dataset

Let us visualize every traffic sign and get to know it's meaning

In [20]:

```
rows = 7
cols = 7
fig=plt.figure(figsize=(25,25))
axes=[]
dict_={'fontweight':"bold"}
for i in range(0,43):
    val=str(format(i,'02d'))
    path="E:\\GTSDB\\TrainIJCNN2013\\" + val + "\\00000.ppm"
    img=cv2.imread(path)
    img=img.astype('float32')
    img=img/255.
    axes.append(fig.add_subplot(rows, cols, i+1) )
    subplot_title=(signnames['SignName'][i])
    axes[-1].set_title(subplot_title,fontdict=dict_)
    plt.imshow(img)

fig.tight_layout()
plt.show()
```

Let us Check One Image Pertaining to each of the four Classes

- From the text file given along with the dataset,it clearly mentions the signs that belong to each of the four above mentioned classes.So let's segregate the traffic signs into the four classes

0 = speed limit 20 (prohibitory)
1 = speed limit 30 (prohibitory)

2 = speed limit 50 (prohibitory)

3 = speed limit 60 (prohibitory)

4 = speed limit 70 (prohibitory)

5 = speed limit 80 (prohibitory)

6 = restriction ends 80 (other)

7 = speed limit 100 (prohibitory)

8 = speed limit 120 (prohibitory)

9 = no overtaking (prohibitory)

10 = no overtaking (trucks) (prohibitory)

11 = priority at next intersection (danger)

12 = priority road (other)

13 = give way (other)

14 = stop (other)

15 = no traffic both ways (prohibitory)

16 = no trucks (prohibitory)

17 = no entry (other)

18 = danger (danger)

19 = bend left (danger)

20 = bend right (danger)

21 = bend (danger)

22 = uneven road (danger)

23 = slippery road (danger)

24 = road narrows (danger)

25 = construction (danger)

26 = traffic signal (danger)

27 = pedestrian crossing (danger)

28 = school crossing (danger)

29 = cycles crossing (danger)

30 = snow (danger)

31 = animals (danger)

32 = restriction ends (other)

33 = go right (mandatory)

34 = go left (mandatory)

35 = go straight (mandatory)

36 = go right or straight (mandatory)

37 = go left or straight (mandatory)

38 = keep right (mandatory)

39 = keep left (mandatory)

40 = roundabout (mandatory)

41 = restriction ends (overtaking) (other)

42 = restriction ends (overtaking (trucks)) (other)

The respective categories consist of the following traffic sign classes: prohibitory = [0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 15, 16] (circular, white ground with red border) mandatory = [33, 34, 35, 36, 37, 38, 39, 40] (circular, blue ground) danger = [11, 18, 19, 20 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31] (triangular, white ground with red border)

In [5]:

```
def img_plot(path):
    img=cv2.imread(path)
    img=img.astype('float32')
    img=img/255.
    return img
```

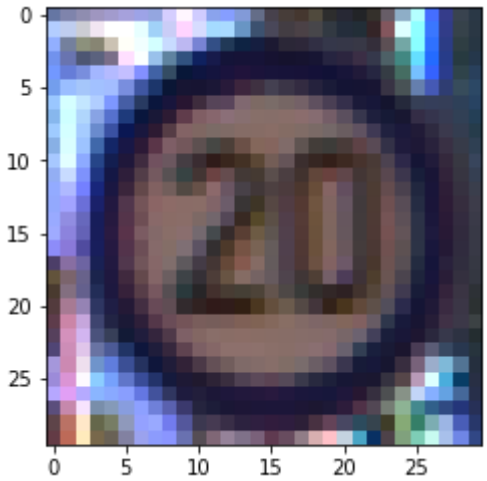
Prohibitory

In [10]:

```
path="E:\\GTSDB\\Traffic Signs\\00\\00000.ppm"
plt.imshow(img_plot(path))
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x20bcfd19580>
```



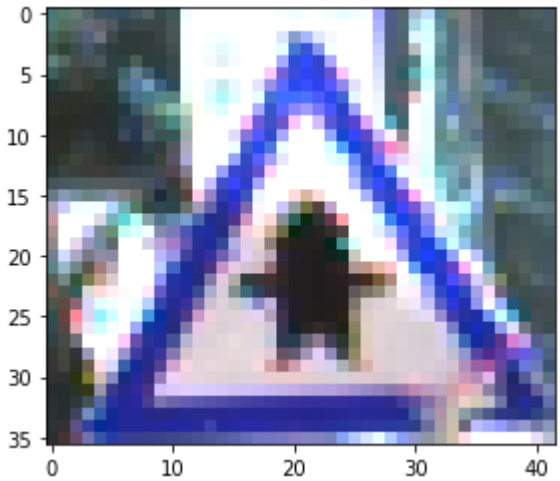
Danger

In [22]:

```
path="E:\\GTSDB\\TrainIJCNN2013\\11\\00000.ppm"
plt.imshow(img_plot(path))
```

Out[22]:

```
<matplotlib.image.AxesImage at 0x20bcff81070>
```



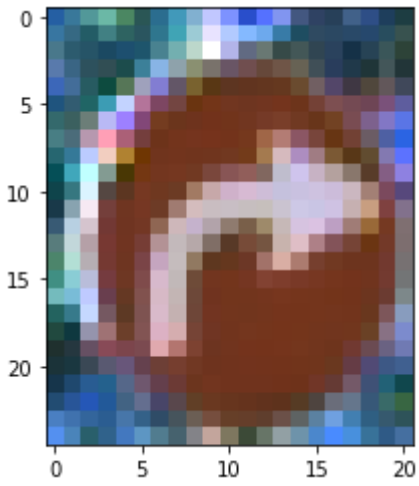
Mandatory

In [23]:

```
path="E:\\GTSDB\\TrainIJCNN2013\\33\\00000.ppm"
plt.imshow(img_plot(path))
```

Out[23]:

```
<matplotlib.image.AxesImage at 0x20bd00ba3a0>
```



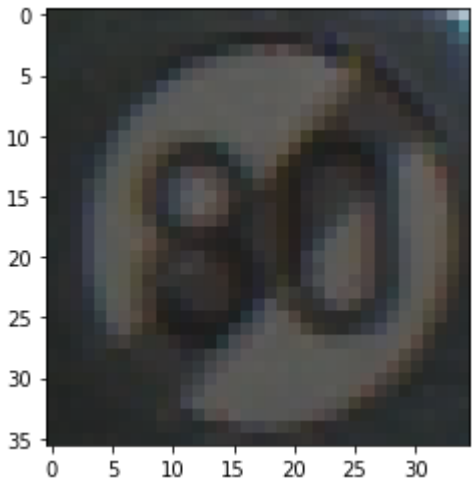
Others

In [24]:

```
path="E:\\GTSDB\\TrainIJCNN2013\\06\\00001.ppm"
plt.imshow(img_plot(path))
```

Out[24]:

```
<matplotlib.image.AxesImage at 0x20bd01ac7f0>
```



Let Us Check the Distribution of Class Label Over the Train Data

In [2]:

```
# Reading txt file with annotations separated by semicolons
# Loading six columns into Pandas dataFrame
# Giving at the same time names to the columns
import pandas as pd
annotation = pd.read_csv('E:\\GTSDB\\gt.txt',
                  names=['ImageID',
                         'XMin',
                         'YMin',
                         'XMax',
                         'YMax',
                         'ClassLabel'],
                  sep=';')
```

In [3]:

```
prohibitory = [0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 15, 16]
mandatory = [33, 34, 35, 36, 37, 38, 39, 40]
danger = [11, 18, 19, 20,21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]
others =[6,12,13,14,17,32,41,42]
```

In [4]:

```
counter={"prohibitory":0,"mandatory":0,"danger":0,"others":0}

for index,row in annotation.iterrows():

    if row['ClassLabel'] in prohibitory:

        counter["prohibitory"]+=1

    elif row['ClassLabel'] in mandatory:

        counter["mandatory"]+=1

    elif row['ClassLabel'] in danger:

        counter["danger"]+=1

    else:

        counter["others"]+=1
```

In [5]:

```
print(counter.values())
```

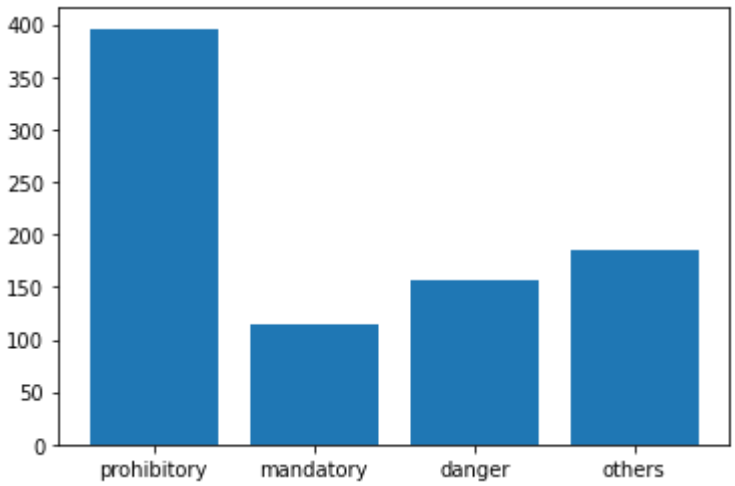
```
dict_values([396, 114, 156, 186])
```

In [8]:

```
plt.bar(counter.keys(),counter.values())
```

Out[8]:

```
<BarContainer object of 4 artists>
```



- **As we can see from the above plot,prohibitory class has highest number of image data.Other 3 classes doesn't differ much**

Data Preprocessing

Preparing data for Keras Batch Generator

In [7]:

```
annotation.head(2)
```

Out[7]:

	ImageID	XMin	YMin	XMax	YMax	ClassLabel
0	00000.ppm	774	411	815	446	11
1	00001.ppm	983	388	1024	432	40

- We might have a question coming up here stating that our training set contains only 600 images but how is there here 851 rows?

- The answer is that each image can contain multiple traffic signs.So a single image can get repeated if it

has multiple traffic signs.

Let us provide the final class_labels by providing the class label pertaining to one of the 4 classes

Assigning Class Labels

In [9]:

```
#Now to the above dataframe we will add some more columns for supporting data in Yolo Forma
annotation['FinalClassID'] = ''
# Getting Final_Class's ID according to the class's labels
# Writing numbers into appropriate column
annotation.loc[annotation['ClassLabel'].isin(prohibitory), 'FinalClassID'] = 0
annotation.loc[annotation['ClassLabel'].isin(danger), 'FinalClassID'] = 1
annotation.loc[annotation['ClassLabel'].isin(mandatory), 'FinalClassID'] = 2
annotation.loc[annotation['ClassLabel'].isin(others), 'FinalClassID'] = 3
```

In [10]:

```
annotation.head(2)
```

Out[10]:

	ImageID	XMin	YMin	XMax	YMax	ClassLabel	FinalClassID
0	00000.ppm	774	411	815	446	11	1
1	00001.ppm	983	388	1024	432	40	2

In [ ]:

```
import os
import cv2
path_dir="E:\\GTSDB\\TrainIJCNN2013"
for current_dir, dirs, files in os.walk(path_dir):
    for f in files:
        #print(f)
        #Check if a file end with .ppm format or not
        if f.endswith('.ppm'):
            img=cv2.imread(os.path.join(path_dir,f))
            image_name = f[:-4]
            save_path = path_dir + '/' + image_name + '.jpg'
            cv2.imwrite(save_path, img)
```

Transforming Test Data from PPM to JPG Format

In [ ]:

```
path_dir="E:\\GTSDB\\TestIJCNN2013"
for current_dir, dirs, files in os.walk(path_dir):
    for f in files:
        #print(f)
        #Check if a file end with .ppm format or not
        if f.endswith('.ppm'):
            img=cv2.imread(os.path.join(path_dir,f))
            image_name = f[:-4]
            save_path = path_dir + '/' + image_name + '.jpg'
            cv2.imwrite(save_path, img)
```

Removing .ppm files from train data

In [ ]:

```
path_dir="E:\\GTSDB\\TrainIJCNN2013"
for current_dir, dirs, files in os.walk(path_dir):
    for f in files:
         if f.endswith('.ppm'):
                os.remove(os.path.join(path_dir,f))
```

Removing .ppm files from test data

In [ ]:

```
path_dir="E:\\GTSDB\\TestIJCNN2013"
for current_dir, dirs, files in os.walk(path_dir):
    for f in files:
         if f.endswith('.ppm'):
                os.remove(os.path.join(path_dir,f))
```

Moving Files That has no annotations in Train Data

In [4]:

```
import shutil,os
for current_dir, dirs, files in os.walk("E:\\GTSDB\\TrainIJCNN2013\\"):
    for file in files:
        if file.endswith(".jpg"):
            temp=file.split('.')[0] + ".ppm"
            if temp in annotation["ImageID"].values:
                print("{} Annotation available".format(file))

            else:
                shutil.move("E:/GTSDB/TrainIJCNN2013/" + file ,"E:/GTSDB/no_annot_train")
```

```
00000.jpg Annotation available
00001.jpg Annotation available
00002.jpg Annotation available
00003.jpg Annotation available
00004.jpg Annotation available
00005.jpg Annotation available
00006.jpg Annotation available
00007.jpg Annotation available
00008.jpg Annotation available
00009.jpg Annotation available
00010.jpg Annotation available
00011.jpg Annotation available
00012.jpg Annotation available
00013.jpg Annotation available
00014.jpg Annotation available
00015.jpg Annotation available
00016.jpg Annotation available
00017.jpg Annotation available
00018.jpg Annotation available
```

In [13]:

```
annotation.to_csv("E:\\GTSDB\\annotation.csv",index=False)
```