Traffic Sign Recognition with CNN

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from PIL import Image
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import accuracy_score
np.random.seed(42)

from matplotlib import style
style.use('fivethirtyeight')
```

Read data

In [2]:
```
data_dir = '../input/gtsrb-german-traffic-sign'
train_path = '../input/gtsrb-german-traffic-sign/Train'
test_path = '../input/gtsrb-german-traffic-sign/'

# Resizing the images to 30x30x3
IMG_HEIGHT = 30
IMG_WIDTH = 30
channels = 3
```

Finding Total Classes

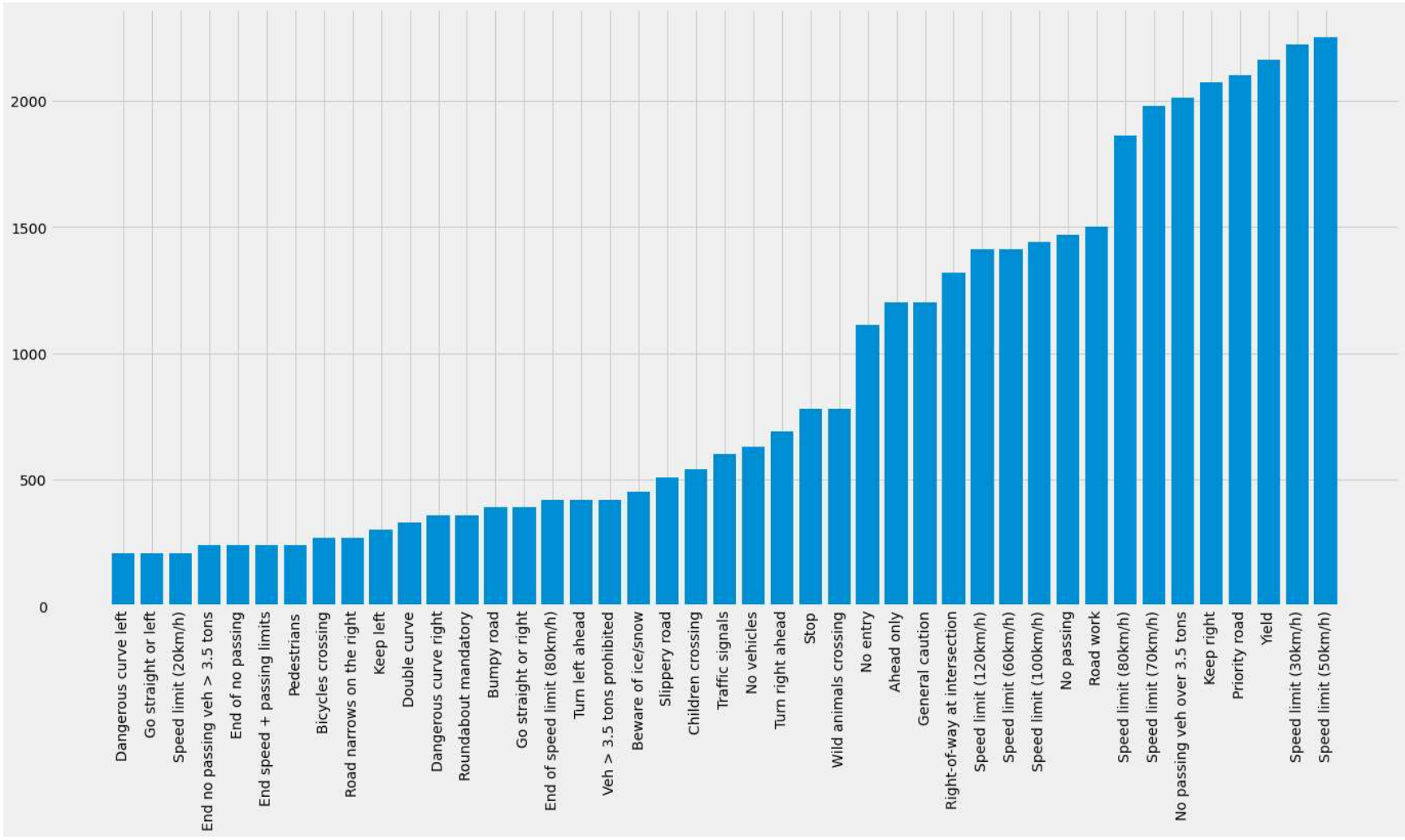
In [3]:
```
NUM_CATEGORIES = len(os.listdir(train_path))
NUM_CATEGORIES
```

Out[3]:
```
43
```

In [4]:
```
# Label Overview
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
            6:'End of speed limit (80km/h)',
            7:'Speed limit (100km/h)',
            8:'Speed limit (120km/h)',
            9:'No passing',
            10:'No passing veh over 3.5 tons',
            11:'Right-of-way at intersection',
            12:'Priority road',
            13:'Yield',
            14:'Stop',
            15:'No vehicles',
            16:'Veh > 3.5 tons prohibited',
            17:'No entry',
            18:'General caution',
            19:'Dangerous curve left',
            20:'Dangerous curve right',
            21:'Double curve',
            22:'Bumpy road',
            23:'Slippery road',
            24:'Road narrows on the right',
            25:'Road work',
            26:'Traffic signals',
            27:'Pedestrians',
            28:'Children crossing',
            29:'Bicycles crossing',
            30:'Beware of ice/snow',
            31:'Wild animals crossing',
            32:'End speed + passing limits',
            33:'Turn right ahead',
            34:'Turn left ahead',
            35:'Ahead only',
            36:'Go straight or right',
            37:'Go straight or left',
            38:'Keep right',
            39:'Keep left',
            40:'Roundabout mandatory',
            41:'End of no passing',
            42:'End no passing veh > 3.5 tons' }
```

Visualizing The Dataset

In [5]:
```
folders = os.listdir(train_path)

train_number = []
class_num = []

for folder in folders:
    train_files = os.listdir(train_path + '/' + folder)
    train_number.append(len(train_files))
    class_num.append(classes[int(folder)])

# Sorting the dataset on the basis of number of images in each class
zipped_lists = zip(train_number, class_num)
sorted_pairs = sorted(zipped_lists)
tuples = zip(*sorted_pairs)
train_number, class_num = [ list(tuple) for tuple in  tuples]

# Plotting the number of images in each class
plt.figure(figsize=(21,10))
plt.bar(class_num, train_number)
plt.xticks(class_num, rotation='vertical')
plt.show()
```



In [6]:
```
# Visualizing 25 random images from test data
import random
from matplotlib.image import imread

test = pd.read_csv(data_dir + '/Test.csv')
imgs = test["Path"].values

plt.figure(figsize=(25,25))

for i in range(1,26):
    plt.subplot(5,5,i)
    random_img_path = data_dir + '/' + random.choice(imgs)
    rand_img = imread(random_img_path)
    plt.imshow(rand_img)
    plt.grid(b=None)
    plt.xlabel(rand_img.shape[1], fontsize = 20)#width of image
    plt.ylabel(rand_img.shape[0], fontsize = 20)#height of image
```

Collecting the Training Data

In [7]:
```
image_data = []
image_labels = []

for i in range(NUM_CATEGORIES):
    path = data_dir + '/Train/' + str(i)
    images = os.listdir(path)

    for img in images:
        try:
            image = cv2.imread(path + '/' + img)
            image_fromarray = Image.fromarray(image, 'RGB')
            resize_image = image_fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
            image_data.append(np.array(resize_image))
            image_labels.append(i)
        except:
            print("Error in " + img)

# Changing the list to numpy array
image_data = np.array(image_data)
image_labels = np.array(image_labels)

print(image_data.shape, image_labels.shape)
```

```
(39209, 30, 30, 3) (39209,)
```

Shuffling the training data

In [8]:
```
shuffle_indexes = np.arange(image_data.shape[0])
np.random.shuffle(shuffle_indexes)
image_data = image_data[shuffle_indexes]
image_labels = image_labels[shuffle_indexes]
```

Splitting the data into train and validation set

In [9]:
```
X_train, X_val, y_train, y_val = train_test_split(image_data, image_labels, test_size=0.3, random_state=42, shuffle=Tru

X_train = X_train/255
X_val = X_val/255
```

```
print("X_train.shape", X_train.shape)
print("X_valid.shape", X_val.shape)
print("y_train.shape", y_train.shape)
print("y_valid.shape", y_val.shape)
```

```
X_train.shape (27446, 30, 30, 3)
X_valid.shape (11763, 30, 30, 3)
y_train.shape (27446,)
y_valid.shape (11763,)
```

One hot encoding the labels

In [10]:
```
y_train = keras.utils.to_categorical(y_train, NUM_CATEGORIES)
y_val = keras.utils.to_categorical(y_val, NUM_CATEGORIES)

print(y_train.shape)
print(y_val.shape)
```

```
(27446, 43)
(11763, 43)
```

Building the model

In [11]:
```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=16, kernel_size=(3,3), activation='relu', input_shape=(IMG_HEIGHT,IMG_WIDTH,channels)),
    keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    keras.layers.Conv2D(filters=128, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Flatten(),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate=0.5),

    keras.layers.Dense(43, activation='softmax')
])
```

In [12]:
```
lr = 0.001
epochs = 30

opt = Adam(lr=lr, decay=lr / (epochs * 0.5))
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

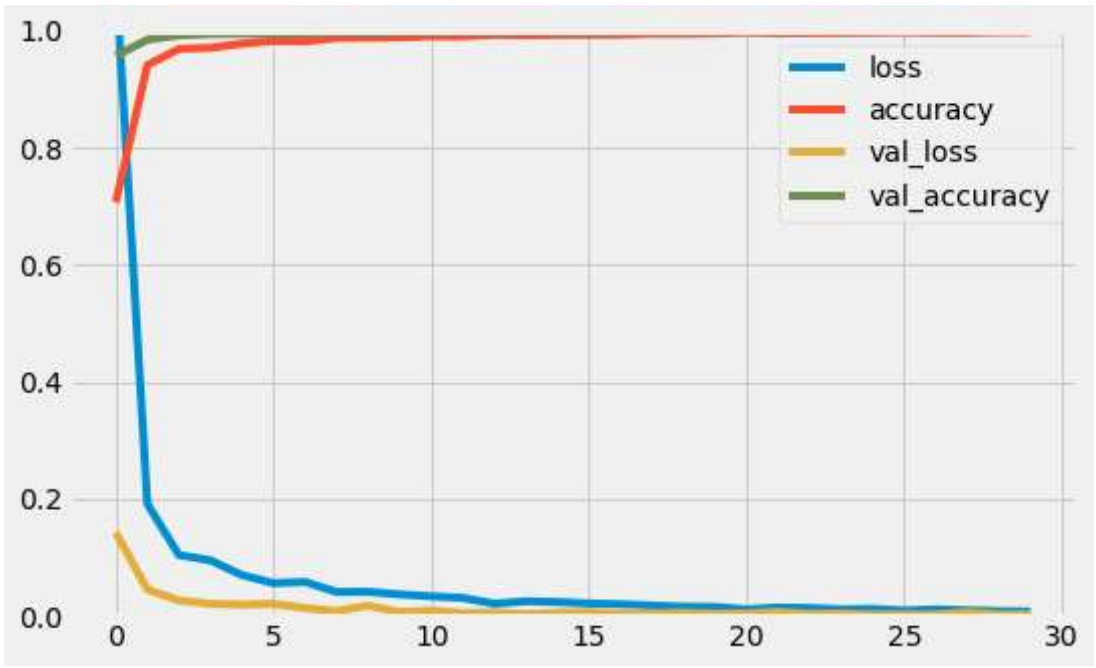
Augmenting the data and training the model

In [13]:
```
aug = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.15,
    horizontal_flip=False,
    vertical_flip=False,
    fill_mode="nearest")

history = model.fit(aug.flow(X_train, y_train, batch_size=32), epochs=epochs, validation_data=(X_val, y_val))
```

```
Epoch 1/30
858/858 [==============================] - 15s 18ms/step - loss: 1.0908 - accuracy: 0.7070 - val_loss: 0.1429 - val_acc
uracy: 0.9566
Epoch 2/30
858/858 [==============================] - 15s 18ms/step - loss: 0.1918 - accuracy: 0.9410 - val_loss: 0.0459 - val_acc
uracy: 0.9843
Epoch 3/30
858/858 [==============================] - 15s 18ms/step - loss: 0.1047 - accuracy: 0.9686 - val_loss: 0.0271 - val_acc
uracy: 0.9917
Epoch 4/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0956 - accuracy: 0.9703 - val_loss: 0.0215 - val_acc
uracy: 0.9940
Epoch 5/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0704 - accuracy: 0.9777 - val_loss: 0.0199 - val_acc
uracy: 0.9946
Epoch 6/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0566 - accuracy: 0.9819 - val_loss: 0.0213 - val_acc
uracy: 0.9934
Epoch 7/30
858/858 [==============================] - 15s 17ms/step - loss: 0.0588 - accuracy: 0.9814 - val_loss: 0.0143 - val_acc
uracy: 0.9950
Epoch 8/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0416 - accuracy: 0.9874 - val_loss: 0.0088 - val_acc
uracy: 0.9975
Epoch 9/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0419 - accuracy: 0.9876 - val_loss: 0.0184 - val_acc
uracy: 0.9943
Epoch 10/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0375 - accuracy: 0.9882 - val_loss: 0.0074 - val_acc
uracy: 0.9977
Epoch 11/30
858/858 [==============================] - 15s 17ms/step - loss: 0.0342 - accuracy: 0.9902 - val_loss: 0.0099 - val_acc
uracy: 0.9967
Epoch 12/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0315 - accuracy: 0.9900 - val_loss: 0.0056 - val_acc
uracy: 0.9985
Epoch 13/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0216 - accuracy: 0.9928 - val_loss: 0.0066 - val_acc
uracy: 0.9980
Epoch 14/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0257 - accuracy: 0.9920 - val_loss: 0.0038 - val_acc
uracy: 0.9990
Epoch 15/30
858/858 [==============================] - 15s 17ms/step - loss: 0.0245 - accuracy: 0.9926 - val_loss: 0.0062 - val_acc
uracy: 0.9981
Epoch 16/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0220 - accuracy: 0.9929 - val_loss: 0.0071 - val_acc
uracy: 0.9980
Epoch 17/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0208 - accuracy: 0.9932 - val_loss: 0.0047 - val_acc
uracy: 0.9987
Epoch 18/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0185 - accuracy: 0.9945 - val_loss: 0.0050 - val_acc
uracy: 0.9987
Epoch 19/30
858/858 [==============================] - 15s 17ms/step - loss: 0.0165 - accuracy: 0.9944 - val_loss: 0.0060 - val_acc
uracy: 0.9985
Epoch 20/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0161 - accuracy: 0.9952 - val_loss: 0.0044 - val_acc
uracy: 0.9986
Epoch 21/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0119 - accuracy: 0.9966 - val_loss: 0.0050 - val_acc
uracy: 0.9988
Epoch 22/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0151 - accuracy: 0.9953 - val_loss: 0.0075 - val_acc
uracy: 0.9978
Epoch 23/30
858/858 [==============================] - 15s 17ms/step - loss: 0.0142 - accuracy: 0.9957 - val_loss: 0.0041 - val_acc
uracy: 0.9991
Epoch 24/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0119 - accuracy: 0.9958 - val_loss: 0.0048 - val_acc
uracy: 0.9987
Epoch 25/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0128 - accuracy: 0.9962 - val_loss: 0.0030 - val_acc
uracy: 0.9994
Epoch 26/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0098 - accuracy: 0.9970 - val_loss: 0.0034 - val_acc
uracy: 0.9991
Epoch 27/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0119 - accuracy: 0.9961 - val_loss: 0.0030 - val_acc
uracy: 0.9992
Epoch 28/30
858/858 [==============================] - 15s 17ms/step - loss: 0.0103 - accuracy: 0.9964 - val_loss: 0.0070 - val_acc
uracy: 0.9983
Epoch 29/30
858/858 [==============================] - 16s 18ms/step - loss: 0.0088 - accuracy: 0.9972 - val_loss: 0.0032 - val_acc
uracy: 0.9992
Epoch 30/30
858/858 [==============================] - 15s 18ms/step - loss: 0.0084 - accuracy: 0.9974 - val_loss: 0.0037 - val_acc
uracy: 0.9989
```

Evaluating the model

In [14]:
```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



Loading the test data and running the predictions

In [15]:
```
test = pd.read_csv(data_dir + '/Test.csv')

labels = test["ClassId"].values
imgs = test["Path"].values

data =[]

for img in imgs:
    try:
        image = cv2.imread(data_dir + '/' +img)
        image_fromarray = Image.fromarray(image, 'RGB')
        resize_image = image_fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
        data.append(np.array(resize_image))
    except:
        print("Error in " + img)
X_test = np.array(data)
X_test = X_test/255

pred = model.predict_classes(X_test)

#Accuracy with the test data
print('Test Data accuracy: ',accuracy_score(labels, pred)*100)
```

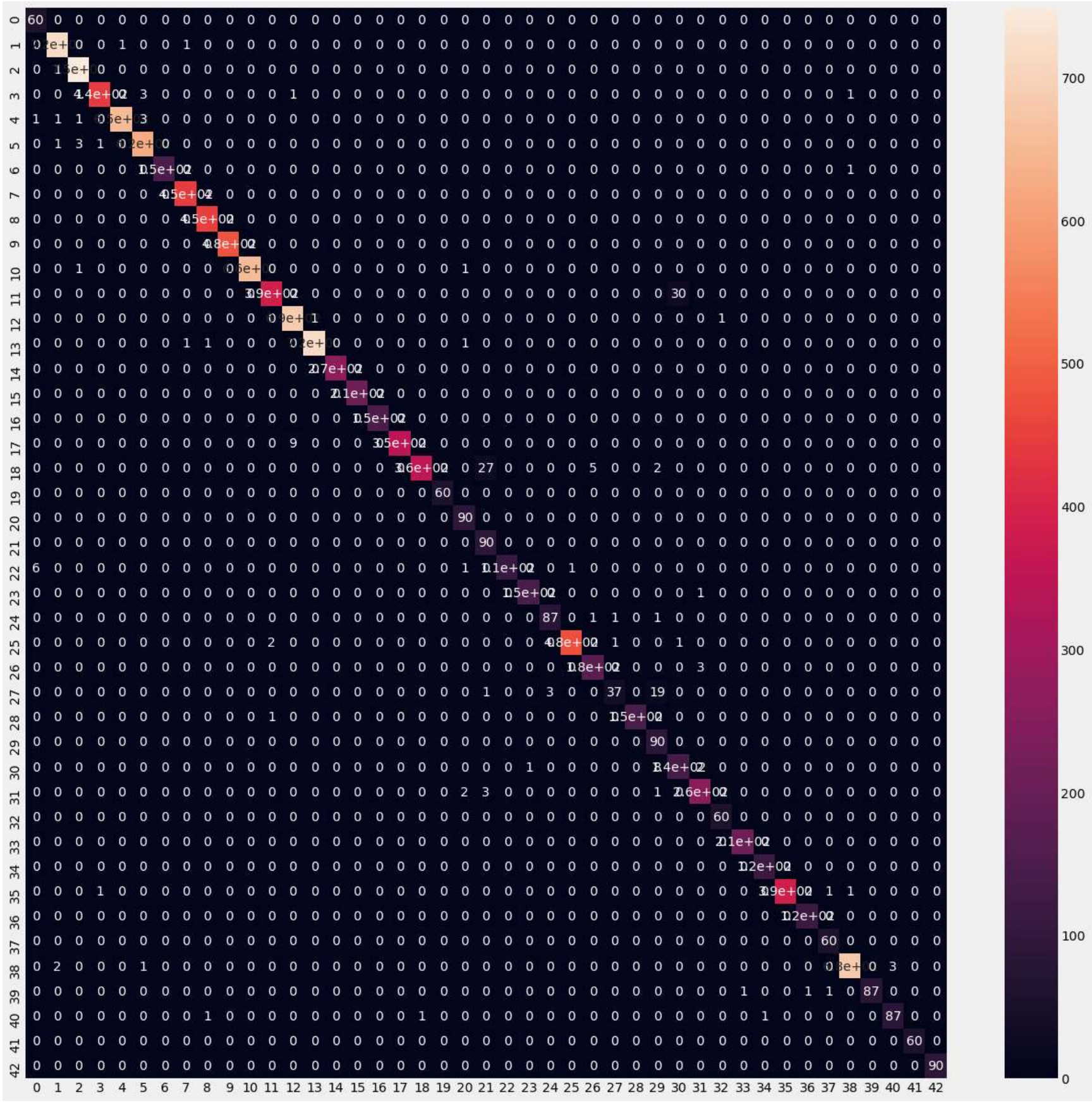
```
Test Data accuracy:  98.57482185273159
```

Confusion matrix

In [16]:
```
from sklearn.metrics import confusion_matrix
cf = confusion_matrix(labels, pred)
```

In [17]:
```
import seaborn as sns
df_cm = pd.DataFrame(cf, index = classes,  columns = classes)
plt.figure(figsize = (20,20))
sns.heatmap(df_cm, annot=True)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7dc22c1d56d0>
```



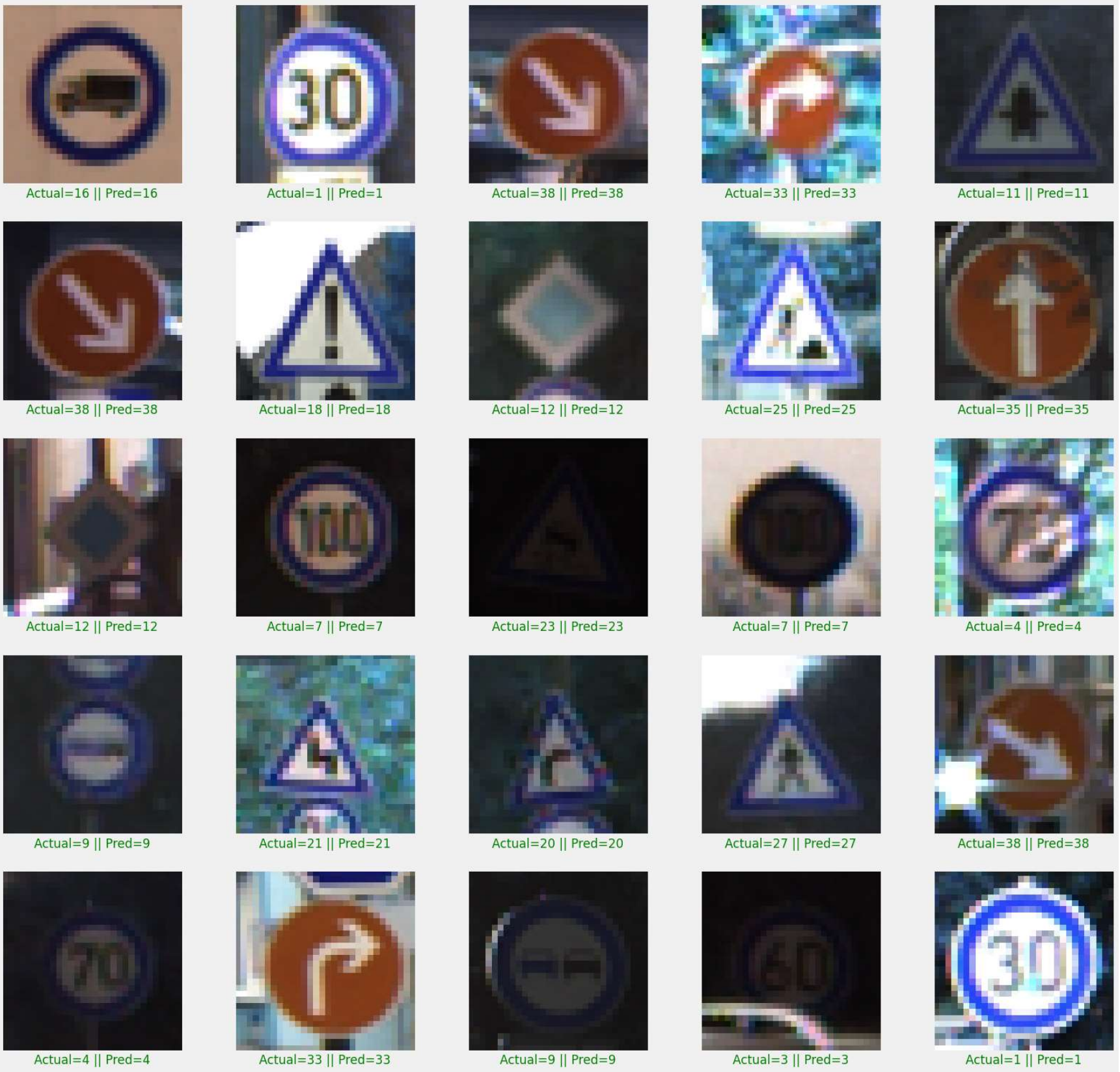
Classification report

In [18]:
```
from sklearn.metrics import classification_report

print(classification_report(labels, pred))
```

	precision	recall	f1-score	support
0	0.90	1.00	0.94	60
1	0.99	1.00	1.00	720
2	0.99	1.00	1.00	750
3	1.00	0.99	0.99	450
4	1.00	0.99	0.99	660
5	0.99	0.99	0.99	630
6	1.00	0.99	1.00	150
7	1.00	0.99	0.99	450
8	0.99	1.00	0.99	450
9	1.00	1.00	1.00	480
10	1.00	1.00	1.00	660
11	0.99	0.93	0.96	420
12	0.99	1.00	0.99	690
13	1.00	1.00	1.00	720
14	1.00	1.00	1.00	270
15	1.00	1.00	1.00	210
16	1.00	1.00	1.00	150
17	1.00	0.97	0.99	360
18	1.00	0.91	0.95	390
19	1.00	1.00	1.00	60
20	0.95	1.00	0.97	90
21	0.74	1.00	0.85	90
22	1.00	0.93	0.97	120
23	0.99	0.99	0.99	150
24	0.97	0.97	0.97	90
25	1.00	0.99	0.99	480
26	0.97	0.98	0.98	180
27	0.95	0.62	0.75	60
28	1.00	0.99	1.00	150
29	0.74	1.00	0.85	90
30	0.82	0.93	0.87	150
31	0.98	0.98	0.98	270
32	0.98	1.00	0.99	60
33	1.00	1.00	1.00	210
34	0.99	1.00	1.00	120
35	1.00	0.99	1.00	390
36	0.99	1.00	1.00	120
37	0.97	1.00	0.98	60
38	1.00	0.99	0.99	690
39	1.00	0.97	0.98	90
40	0.97	0.97	0.97	90
41	1.00	1.00	1.00	60
42	1.00	1.00	1.00	90
accuracy			0.99	12630
macro avg	0.97	0.98	0.97	12630
weighted avg	0.99	0.99	0.99	12630

Predictions on Test Data

In [19]:
```
plt.figure(figsize = (25, 25))

start_index = 0
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    prediction = pred[start_index + i]
    actual = labels[start_index + i]
    col = 'g'
    if prediction != actual:
        col = 'r'
    plt.xlabel('Actual={} || Pred={}'.format(actual, prediction), color = col)
    plt.imshow(X_test[start_index + i])
plt.show()
```


Actual=16 || Pred=16
Actual=1 || Pred=1
Actual=38 || Pred=38
Actual=33 || Pred=33
Actual=11 || Pred=11
Actual=38 || Pred=38
Actual=18 || Pred=18
Actual=12 || Pred=12
Actual=25 || Pred=25
Actual=35 || Pred=35
Actual=12 || Pred=12
Actual=7 || Pred=7
Actual=23 || Pred=23
Actual=7 || Pred=7
Actual=4 || Pred=4
Actual=9 || Pred=9
Actual=21 || Pred=21
Actual=20 || Pred=20
Actual=27 || Pred=27
Actual=38 || Pred=38
Actual=4 || Pred=4
Actual=33 || Pred=33
Actual=9 || Pred=9
Actual=3 || Pred=3
Actual=1 || Pred=1