Traffic_Sign

April 17, 2021

Name: Ventrapragada Sai Shravani
PRN:17070123120
Batch: G-5 (2017-21)

Import libraries

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import keras
import cv2
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Dense
from keras.utils.np_utils import to_categorical
from keras.layers import Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
import pickle
import random
import pandas as pd
```

Import Dataset

[2]:
```
!git clone https://bitbucket.org/jadslim/german-traffic-signs
```

```
Cloning into 'german-traffic-signs'...
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
```

[3]:
```
with open('german-traffic-signs/train.p','rb') as f:
    train_data = pickle.load(f)
with open('german-traffic-signs/valid.p','rb') as f:
    val_data = pickle.load(f)
with open('german-traffic-signs/test.p','rb') as f:
    test_data = pickle.load(f)
```

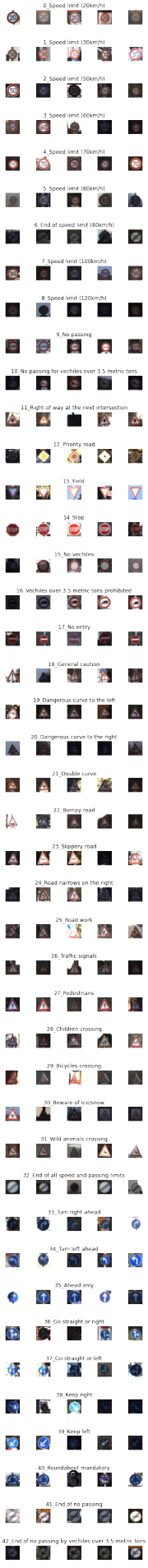
[4]:
```
X_train, y_train = train_data['features'], train_data['labels']
X_val, y_val = val_data['features'], val_data['labels']
X_test, y_test = test_data['features'], test_data['labels']
```

[5]:
```
print(X_train.shape)
print(X_val.shape)
print(X_test.shape)
```

```
(34799, 32, 32, 3)
(4410, 32, 32, 3)
(12630, 32, 32, 3)
```

Data Visulisation

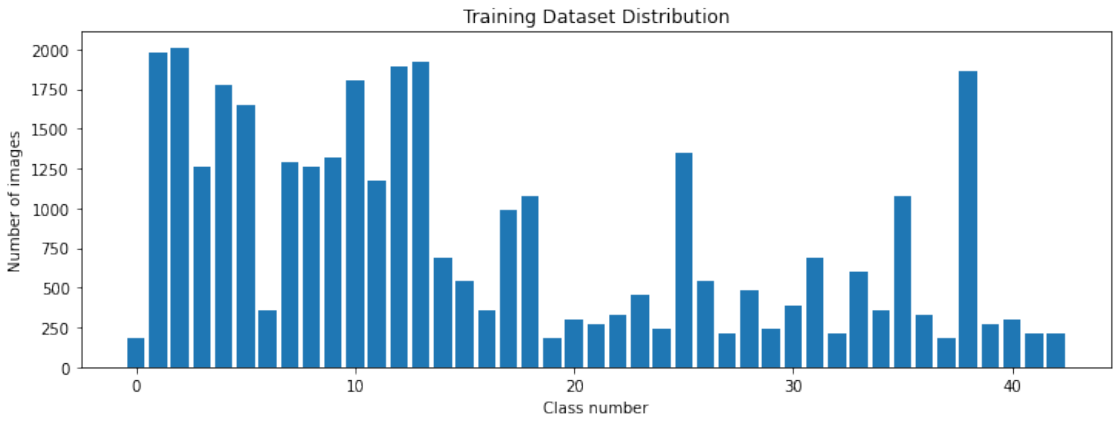
[6]:
```
data = pd.read_csv('german-traffic-signs/signnames.csv')

num_of_samples = []

cols = 5
num_classes = 43

fig, axs = plt.subplots(nrows = num_classes, ncols = cols, figsize = (5, 50))
fig.tight_layout()

for i in range(cols):
    for j, row in data.iterrows():
        x_selected = X_train[y_train == j]
        axs[j][i].imshow(x_selected[random.randint(0, (len(x_selected)-1)), :, :
 ↪], cmap = plt.get_cmap("gray"))
        axs[j][i].axis("off")
        if i == 2:
            axs[j][i].set_title(str(j) + "_" + row["SignName"])
            num_of_samples.append(len(x_selected))
```


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 for vechiles over 3.5 metric tons
11_Right-of-way at the next intersection
12_Priority road
13_Yield
14_Stop
15_No vechiles
16_Vechiles over 3.5 metric tons prohibited
17_No entry
18_General caution
19_Dangerous curve to the left
20_Dangerous curve to the 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 of all speed and 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 of no passing by vechiles over 3.5 metric tons

[7]:
```
# Distribution of classes
print(num_of_samples)
plt.figure(figsize = (12, 4))
plt.bar(range(0, num_classes), num_of_samples)
plt.title("Training Dataset Distribution")
plt.xlabel("Class number")
plt.ylabel("Number of images")
```

```
[180, 1980, 2010, 1260, 1770, 1650, 360, 1290, 1260, 1320, 1800, 1170, 1890,
1920, 690, 540, 360, 990, 1080, 180, 300, 270, 330, 450, 240, 1350, 540, 210,
480, 240, 390, 690, 210, 599, 360, 1080, 330, 180, 1860, 270, 300, 210, 210]
```

[7]:
```
Text(0, 0.5, 'Number of images')
```



Data Augmentation for making images huges or add 42/ 43 more images

[8]:
```
plt.imshow(X_train[1000])
plt.axis('off')
print(X_train[1000].shape)
print(y_train[1000])
```

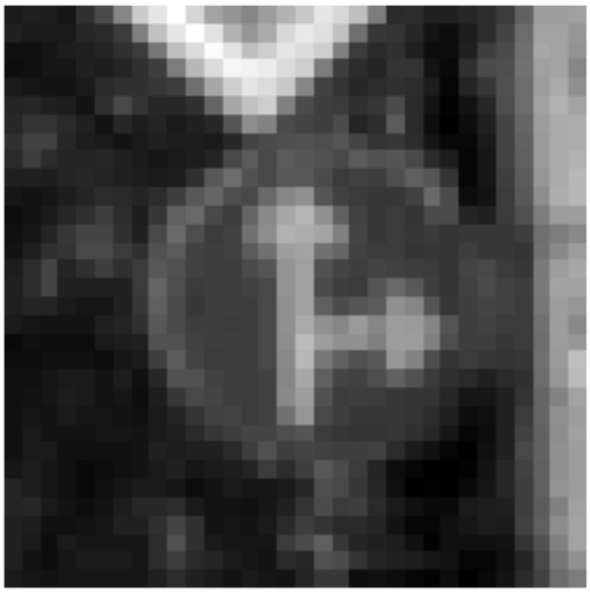
```
(32, 32, 3)
36
```

Pre-processing has to be done

[9]:
```
# Color does not matters here shapes and sign matters
def grayscale(img):
    image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    plt.axis('off')
    return image
```

[10]:
```
img = grayscale(X_train[1000])
plt.imshow(img, cmap = 'gray')
print(img.shape)
```

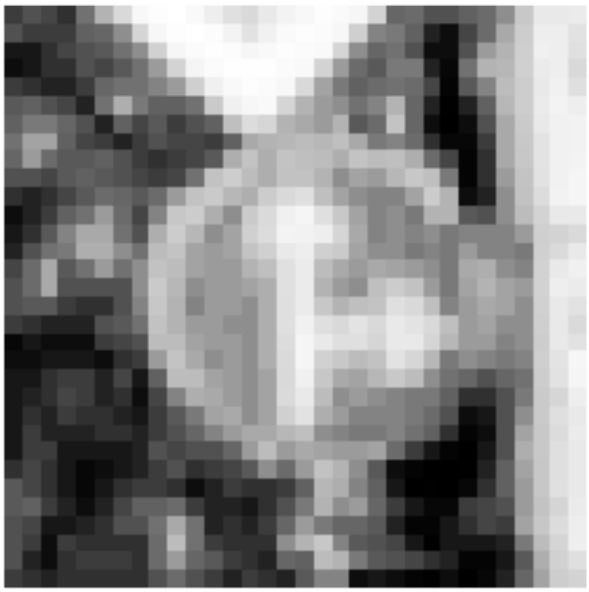
```
(32, 32)
```



[11]:
```
# Histogram equalisation
def equalize(img):
    img = cv2.equalizeHist(img)
    return img
```

[12]:
```
img = equalize(img)
plt.imshow(img, cmap = 'gray')
plt.axis('off')
print(img.shape)
```

```
(32, 32)
```



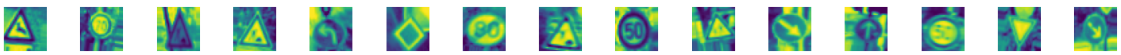
[13]:
```
# Function to preprocess the images in bulk and to normalise it's values
def preprocessing(img):
    img = grayscale(img)
    img = equalize(img)
    img = img/255
    return img
```

[14]:
```
X_train = np.array(list(map(preprocessing, X_train)))
X_val = np.array(list(map(preprocessing, X_val)))
X_test = np.array(list(map(preprocessing, X_test)))
```

[15]:
```
X_train = X_train.reshape(34799, 32, 32, 1) # 1- Section height weight
X_val = X_val.reshape(4410, 32, 32, 1)
X_test = X_test.reshape(12630, 32, 32, 1)
```

[16]:
```
from keras.preprocessing.image import ImageDataGenerator

# Increasing the zoom range height range and width range

datagen = ImageDataGenerator(width_shift_range = 0.1,
                  height_shift_range = 0.1,
                   zoom_range = 0.2,
                  shear_range = 0.1,
                  rotation_range = 10)

datagen.fit(X_train)
```

[17]:
```
# If you directly pass 22 the memory gets occupied completely

batches = datagen.flow(X_train, y_train, batch_size = 20)
X_batch, y_batch = next(batches)

fig, axs = plt.subplots(1, 15, figsize = (20, 5))
fig.tight_layout()

for i in range(15):
  axs[i].imshow(X_batch[i].reshape(32, 32))
  axs[i].axis('off')
```



[18]:
```
y_train = to_categorical(y_train, 43)
y_val = to_categorical(y_val, 43)
y_test = to_categorical(y_test, 43)
```

[19]:
```
def neural_model():
    model = Sequential()
    model.add(Conv2D(60, (5, 5), input_shape = (32, 32, 1), activation =␣
 ↪'relu'))
    model.add(Conv2D(60, (5, 5), input_shape = (32, 32, 1), activation =␣
 ↪'relu'))
    model.add(MaxPooling2D(pool_size = (2,2)))

    model.add(Conv2D(30, (3, 3), activation = 'relu'))
    model.add(Conv2D(30, (3, 3), activation = 'relu'))
```

```
    model.add(MaxPooling2D(pool_size = (2, 2)))

    #model.add(Dropout(0.5))

    model.add(Flatten())
    model.add(Dense(500, activation = 'relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation = 'softmax'))
    # Learning rate is gradient descent
    model.compile(Adam(lr = 0.001), loss = 'categorical_crossentropy', metrics
 ↪= ['accuracy'])
    return model
```

[20]:
```
model = neural_model()
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 60)        1560
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 60)        90060
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 60)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 30)        16230
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 30)          8130
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 30)          0
_________________________________________________________________
flatten (Flatten)            (None, 480)               0
_________________________________________________________________
dense (Dense)                (None, 500)               240500
_________________________________________________________________
dropout (Dropout)            (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 43)                21543
=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
_________________________________________________________________
None
```

Training the model

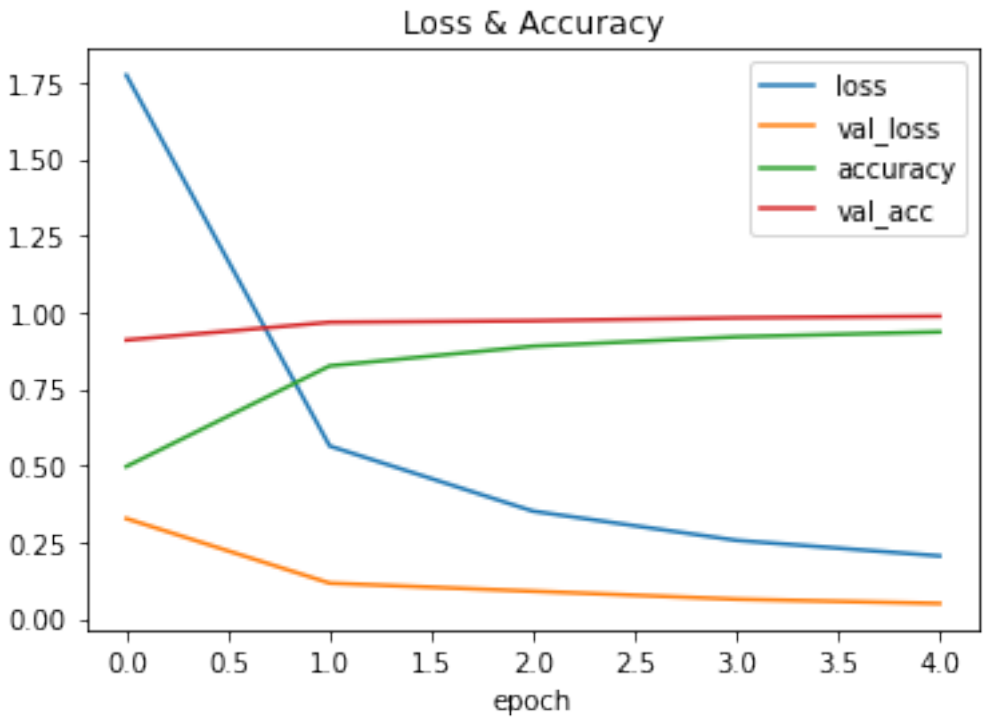
[21]:
```
history = model.fit_generator(datagen.flow(X_train, y_train, batch_size = 50),␣
↪epochs = 5, validation_data =(X_val, y_val), shuffle = 1)
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/training.py:1844: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/5
696/696 [==============================] - 46s 19ms/step - loss: 2.6127 -
accuracy: 0.2875 - val_loss: 0.3267 - val_accuracy: 0.9095
Epoch 2/5
696/696 [==============================] - 12s 18ms/step - loss: 0.6485 -
accuracy: 0.7976 - val_loss: 0.1158 - val_accuracy: 0.9673
Epoch 3/5
696/696 [==============================] - 13s 18ms/step - loss: 0.3807 -
accuracy: 0.8780 - val_loss: 0.0894 - val_accuracy: 0.9723
Epoch 4/5
696/696 [==============================] - 13s 18ms/step - loss: 0.2737 -
accuracy: 0.9130 - val_loss: 0.0634 - val_accuracy: 0.9819
Epoch 5/5
696/696 [==============================] - 13s 18ms/step - loss: 0.2106 -
accuracy: 0.9326 - val_loss: 0.0495 - val_accuracy: 0.9871
```

Evaluating the model

[22]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['loss', 'val_loss', 'accuracy', 'val_acc'])
plt.title('Loss & Accuracy')
plt.xlabel('epoch')
```

[22]:
```
Text(0.5, 0, 'epoch')
```



[23]:
```
score = model.evaluate(X_test, y_test, verbose = 1)
print('Test Score', score[0])
print('Test Accuracy', score[1])
```

```
395/395 [==============================] - 1s 3ms/step - loss: 0.1403 -
accuracy: 0.9641
Test Score 0.1402709037065506
Test Accuracy 0.9640538692474365
```

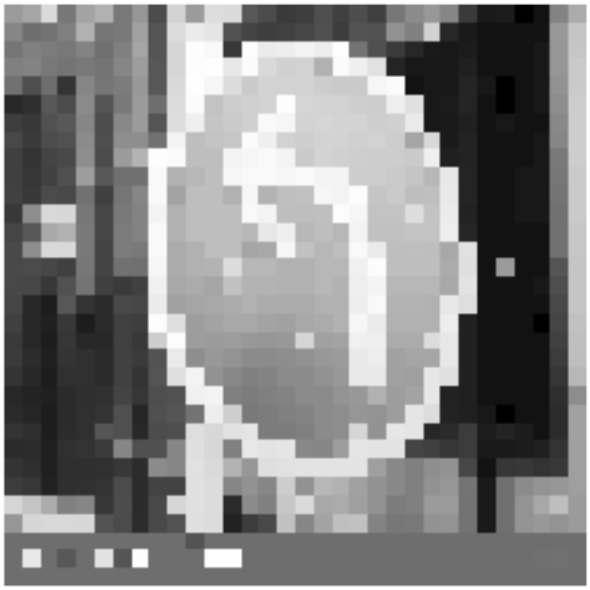
[24]:
```
import requests
from PIL import Image
 # dowlanding the new image
url = 'https://c8.alamy.com/comp/A0RX23/
 ↪cars-and-automobiles-must-turn-left-ahead-sign-A0RX23.jpg'
r = requests.get(url, stream=True)
image = Image.open(r.raw)
plt.axis('off')
plt.imshow(image, cmap=plt.get_cmap('gray'))
```

[24]:
```
<matplotlib.image.AxesImage at 0x7f2755b08c50>
```

Preprocessing images

[25]:
```
img = np.asarray(image)
img = cv2.resize(img, (32, 32))
img = preprocessing(img)
plt.imshow(img, cmap = plt.get_cmap('gray'))
print(img.shape)
img = img.reshape(1, 32, 32, 1) #1 image 32 height and width and 1 channel
```

```
(32, 32)
```



[26]:
```
prediction = str(model.predict_classes(img))

prediction = prediction[1:-1]

##print("predicted sign: "+ prediction )
```

```
/usr/local/lib/python3.7/dist-
packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning:
`model.predict_classes()` is deprecated and will be removed after 2021-01-01.
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model
does multi-class classification   (e.g. if it uses a `softmax` last-layer
activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does
binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

[27]:
```
pred = int(prediction)
plt.imshow(image)
plt.axis('off')

for num, name in data.iteritems():
  name = name.values
  print("predicted sign: "+ str(name[pred]))
```

```
predicted sign: 34
predicted sign: Turn left ahead
```

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Traffic_Sign.ipynb')
```

```
--2021-04-17 06:48:44--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: colab_pdf.py

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2021-04-17 06:48:44 (44.8 MB/s) - colab_pdf.py saved [1864/1864]

Mounted at /content/drive/

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```