```
!pip install -q keras
```

```
!pip install tensorflow
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (2.8
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: absl-py>=0.4.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/dist
Requirement already satisfied: tensorboard<2.9,>=2.8 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.7/di
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: flatbuffers>=1.12 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: gast>=0.2.1 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/py
Requirement already satisfied: tensorflow-estimator<2.9,>=2.8 in /usr/local/lib/python3
Requirement already satisfied: libclang>=9.0.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: keras<2.9,>=2.8.0rc0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/p
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (i
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
import keras
import cv2
from keras.models import Sequential
# from keras.optimizers import Adam
from tensorflow.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
```

```
np.random.seed(0)
```

Importing Data

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

```
Cloning into 'german-traffic-signs'...
Unpacking objects: 100% (6/6), done.
```

```
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)
```

```
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']
```

```
print(X_train.shape)
print(X_val.shape)
print(X_test.shape)
```

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

```
assert(X_train.shape[0] == y_train.shape[0]), "The number of images is not equal to the numbe
assert(X_val.shape[0] == y_val.shape[0]), "The number of images is not equal to the number of
assert(X_test.shape[0] == y_test.shape[0]), "The number of images is not equal to the number
assert(X_train.shape[1:] == (32, 32, 3)), "The dimensions of the image is not 32*32*3"
assert(X_val.shape[1:] == (32, 32, 3)), "The dimensions of the image is not 32*32*3"
```

```
assert(X_test.shape[1:] == (32, 32, 3)), "The dimensions of the image is not 32*32*3"
```

Data Visualisation

```
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
        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)

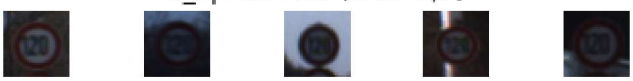
4_Speed limit (70km/h)

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

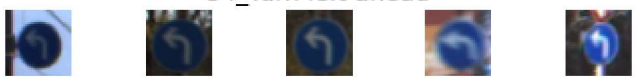
30_Beware of ice/snow

31_Wild animals crossing

32_End of all speed and passing limits

33_Turn right ahead

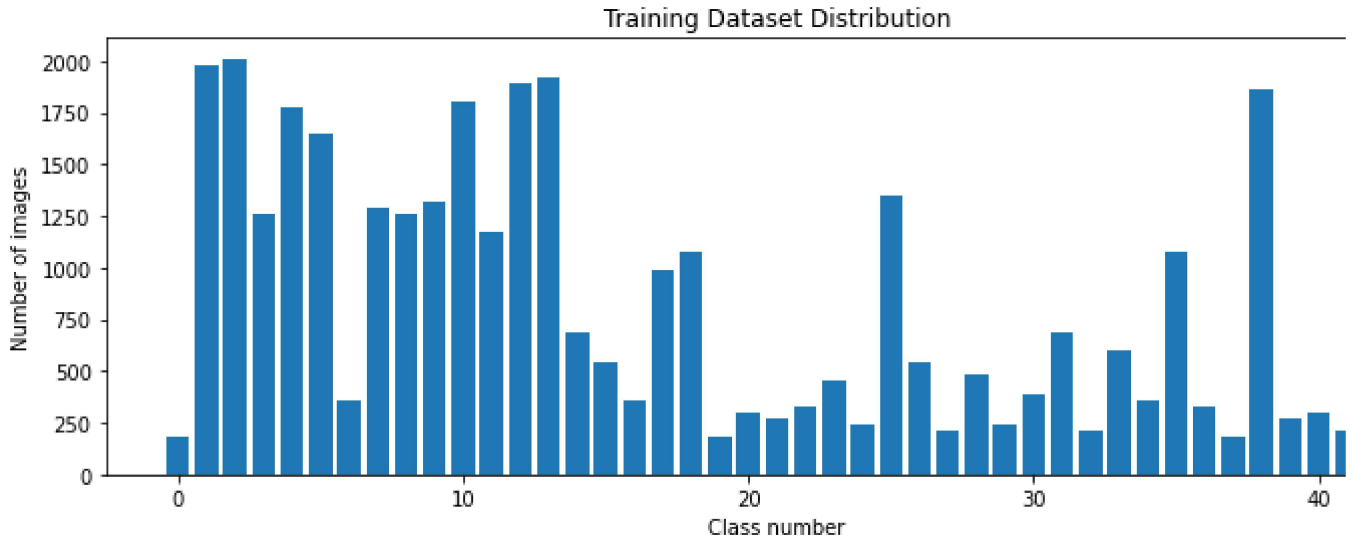
34_Turn left ahead



35_Ahead only


```
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")
```

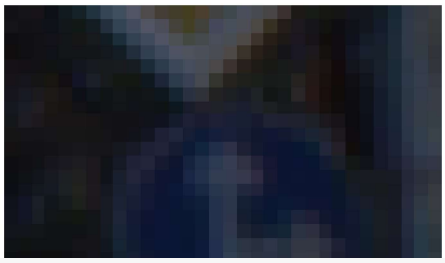
```
[180, 1980, 2010, 1260, 1770, 1650, 360, 1290, 1260, 1320, 1800, 1170, 1890, 1920, 690,
Text(0, 0.5, 'Number of images')
```



Data Preprocessing

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

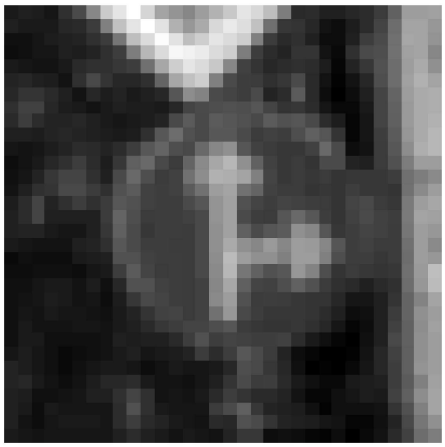
```
(32, 32, 3)
36
```



```
def grayscale(img):
    image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    plt.axis('off')
    return image
```

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

```
(32, 32)
```



```
def equalize(img):
    img = cv2.equalizeHist(img)
    return img
```

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

```
(32, 32)
```

```
def preprocessing(img):
    img = grayscale(img)
    img = equalize(img)
    img = img/255
    return img
```

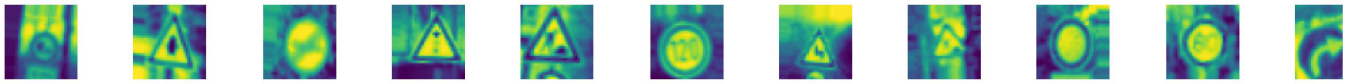
```
X_train = np.array(list(map(preprocessing, X_train)))
X_val = np.array(list(map(preprocessing, X_val)))
X_test = np.array(list(map(preprocessing, X_test)))
```

```
X_train = X_train.reshape(34799, 32, 32, 1)
X_val = X_val.reshape(4410, 32, 32, 1)
X_test = X_test.reshape(12630, 32, 32, 1)
```

```
from keras.preprocessing.image import ImageDataGenerator

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)
```

```
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')
```



```
y_train = to_categorical(y_train, 43)
y_val = to_categorical(y_val, 43)
y_test = to_categorical(y_test, 43)
```

Neural Network

```
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'))
    model.compile(Adam(lr = 0.001), loss = 'categorical_crossentropy', metrics = ['accuracy']
    return model
```

```
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 (MaxPooling  (None, 4, 4, 30)         0
 2D)

 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
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The
  super(Adam, self).__init__(name, **kwargs)
```

```
history = model.fit_generator(datagen.flow(X_train, y_train, batch_size = 50), steps_per_epoc
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_
  """Entry point for launching an IPython kernel.
Epoch 1/10
 694/2000 [=========>....................] - ETA: 22s - loss: 1.6960 - accuracy: 0.5183
2000/2000 [==============================] - 25s 6ms/step - loss: 1.6932 - accuracy: 0.5
```

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

```
395/395 [==============================] - 2s 4ms/step - loss: 0.3881 - accuracy: 0.878
Test Score 0.38807550072669983
Test Accuracy 0.8783056139945984
```

Testing

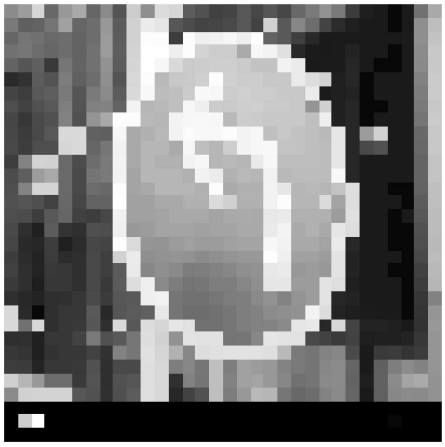
```
import requests
from PIL import Image
url = 'https://c8.alamy.com/comp/A0RX23/cars-and-automobiles-must-turn-left-ahead-sign-A0RX23
r = requests.get(url, stream=True)
```

```
image = Image.open(r.raw)
plt.axis('off')
plt.imshow(image, cmap=plt.get_cmap('gray'))
```

```
<matplotlib.image.AxesImage at 0x7fdb1ade2610>
```


```
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)
```

```
(32, 32)
```



```
prediction = str(model.predict_classes(img))

prediction = prediction[1:-1]

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

```
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: 12
predicted sign: Priority road
```