Traffic Sign Recognition using CNN

In [ ]:
```
from google.colab import files
files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

Out[1]:
```
{'kaggle.json': b'{"username":"gadeakash","key":"46f7761c760a7b4d050ea2375055156f"}'}
```

In [ ]:
```
!pip install -q kaggle
```

In [ ]:
```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle

!chmod 600 ~/.kaggle/kaggle.json
```

In [ ]:
```
!mkdir traffic_sign_dataset
%cd traffic_sign_dataset
```

```
/content/traffic_sign_dataset
```

In [ ]:
```
!kaggle datasets download meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
%cd ..
```

```
Downloading gtsrb-german-traffic-sign.zip to /content/traffic_sign_dataset
 98% 599M/612M [00:08<00:00, 102MB/s]
100% 612M/612M [00:08<00:00, 77.4MB/s]
/content
```

In [ ]:
```
!unzip traffic_sign_dataset/gtsrb-german-traffic-sign.zip -d traffic_sign_dataset
!rm traffic_sign_dataset/gtsrb-german-traffic-sign.zip
!rm -rf traffic_sign_dataset/Meta
!rm -rf traffic_sign_dataset/meta
!rm -rf traffic_sign_dataset/test
!rm -rf traffic_sign_dataset/train
!rm traffic_sign_dataset/Meta.csv
```

```
Streaming output truncated to the last 5000 lines.
  inflating: traffic_sign_dataset/train/5/00005_00053_00010.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00011.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00012.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00013.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00014.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00015.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00016.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00017.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00018.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00019.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00020.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00021.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00022.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00023.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00024.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00025.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00026.png
  inflating: traffic_sign_dataset/train/5/00005_00053_00027.png
```

In [ ]:
```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.image import imread
import random
from PIL import Image
```

In [ ]:
```
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
```

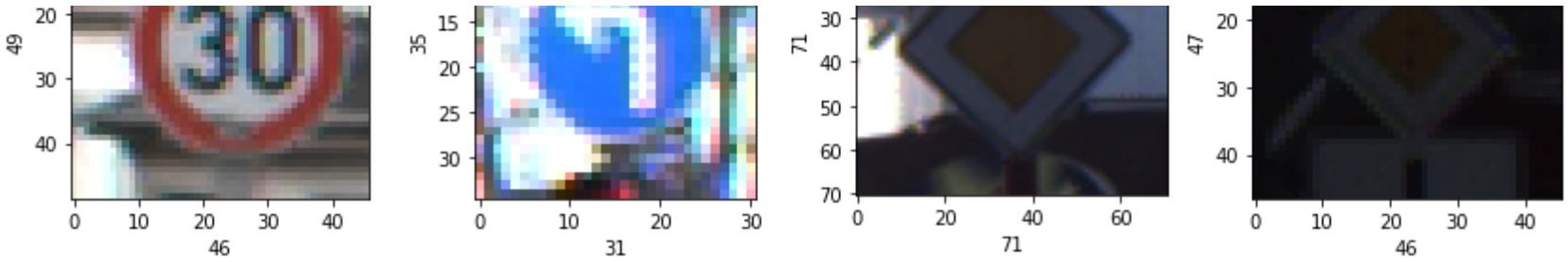
In [ ]:
```
from tensorflow.keras.utils import to_categorical
```

In [ ]:
```
from tensorflow.keras.layers import Dense, Flatten, Dropout, Conv2D, MaxPool2D
```

In [ ]:

```
plt.figure(figsize=(12,12))
path = "traffic_sign_dataset/Test"
for i in range(1,17):
  plt.subplot(4,4,i)
  plt.tight_layout()
  rand_img = imread(path +'/'+ random.choice(sorted(os.listdir(path))))
  plt.imshow(rand_img)
  plt.xlabel(rand_img.shape[1], fontsize=10)
  plt.ylabel(rand_img.shape[0], fontsize=10)
```


STOP



In [ ]:
```
images=[]
label_id = []

for i in range(43):
  labels='traffic_sign_dataset/Train' + '/{0}'.format(i)
  image_path = os.listdir(labels)
  for x in image_path:
    img = Image.open(labels + '/' + x)
    img = img.resize((50,50))
    img = np.array(img)
    images.append(img)
    label_id.append(i)
```

In [ ]:
```
images = np.array(images)
images = images/255
#1 byte = 8 bits.
```

In [ ]:
```
label_id = np.array(label_id)
label_id.shape
```

Out[14]: (39209,)

In [ ]:
```
images.shape
```

Out[15]: (39209, 50, 50, 3)

In [ ]:
```
#Train-test split:
x_train,x_val,y_train,y_val = train_test_split(images, label_id, test_size=0.2,random_state=42)
```

In [ ]:
```
#Fabrication of CNN:
model = Sequential()

#Convolutional Layer Code:
model.add(Conv2D(filters=64,kernel_size=(3,3),input_shape=x_train.shape[1:],activation='relu',padding='same'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.5))


#Pooling Layer Code:
model.add(Conv2D(filters=64, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.5))

#Fully Connected Layer Code: Classification Layer Code
model.add(Flatten()) #This process converts your 4-D frame into 1-D. To input the data into next layer.
model.add(Dense(128, activation='relu')) #f(x)=max(0,x)
model.add(Dropout(0.5)) #This is used for the understanding that the N.N needs to contain several neurons.
model.add(Dense(43, activation='softmax'))
```

In [ ]:
```
images.ndim
```

Out[18]:
```
4
```

In [ ]:
```
#Optimizing a Neural Network:

model.compile(loss= 'sparse_categorical_crossentropy',optimizer='adam',metrics = ['accuracy'])
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 50, 50, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 25, 25, 64)       0
 )

 dropout (Dropout)           (None, 25, 25, 64)        0

 conv2d_1 (Conv2D)           (None, 23, 23, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 11, 11, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 11, 11, 64)        0

 flatten (Flatten)           (None, 7744)              0

 dense (Dense)               (None, 128)               991360

 dropout_2 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 43)                5547

=================================================================
Total params: 1,035,627
Trainable params: 1,035,627
Non-trainable params: 0
_________________________________________________________________
```

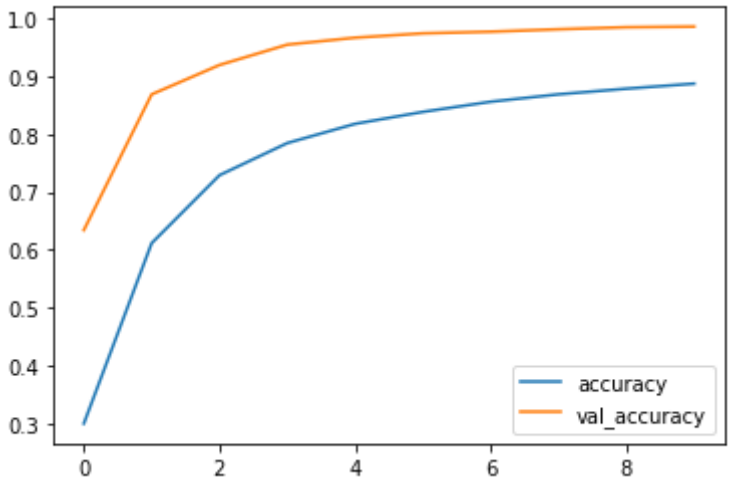
In [ ]:
```
model.fit(x_train,y_train,epochs=10,batch_size=128,validation_data=(x_val,y_val),verbose=2)
```

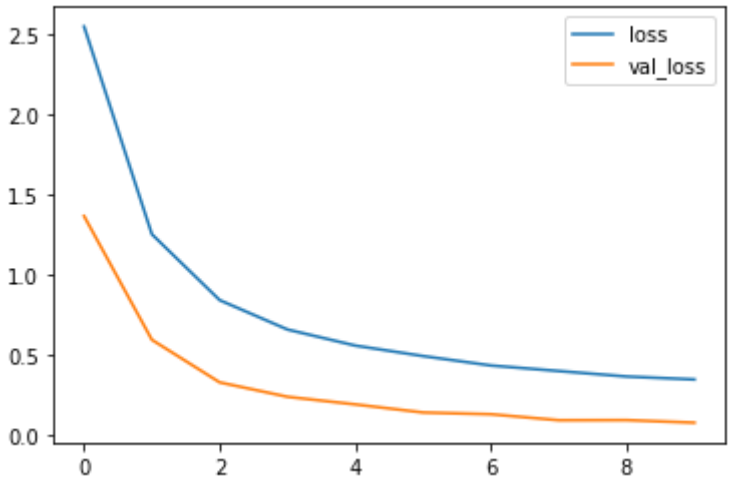
```
Epoch 1/10
246/246 - 149s - loss: 2.5440 - accuracy: 0.3000 - val_loss: 1.3632 - val_accuracy: 0.6341 - 149s/epoch - 607ms/step
Epoch 2/10
246/246 - 148s - loss: 1.2500 - accuracy: 0.6116 - val_loss: 0.5925 - val_accuracy: 0.8687 - 148s/epoch - 602ms/step
Epoch 3/10
246/246 - 147s - loss: 0.8395 - accuracy: 0.7290 - val_loss: 0.3272 - val_accuracy: 0.9194 - 147s/epoch - 599ms/step
Epoch 4/10
246/246 - 150s - loss: 0.6565 - accuracy: 0.7845 - val_loss: 0.2374 - val_accuracy: 0.9546 - 150s/epoch - 609ms/step
Epoch 5/10
246/246 - 150s - loss: 0.5564 - accuracy: 0.8177 - val_loss: 0.1900 - val_accuracy: 0.9666 - 150s/epoch - 610ms/step
Epoch 6/10
246/246 - 151s - loss: 0.4911 - accuracy: 0.8384 - val_loss: 0.1395 - val_accuracy: 0.9741 - 151s/epoch - 615ms/step
Epoch 7/10
246/246 - 157s - loss: 0.4324 - accuracy: 0.8560 - val_loss: 0.1287 - val_accuracy: 0.9769 - 157s/epoch - 638ms/step
Epoch 8/10
246/246 - 151s - loss: 0.3979 - accuracy: 0.8688 - val_loss: 0.0915 - val_accuracy: 0.9811 - 151s/epoch - 614ms/step
Epoch 9/10
246/246 - 147s - loss: 0.3637 - accuracy: 0.8785 - val_loss: 0.0922 - val_accuracy: 0.9847 - 147s/epoch - 599ms/step
Epoch 10/10
246/246 - 146s - loss: 0.3454 - accuracy: 0.8872 - val_loss: 0.0762 - val_accuracy: 0.9857 - 146s/epoch - 592ms/step
```

Out[21]:
```
<keras.callbacks.History at 0x7f988e89d590>
```

In [ ]:
```
evaluation = pd.DataFrame(model.history.history)
evaluation[['accuracy','val_accuracy']].plot()
evaluation[['loss','val_loss']].plot()
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f988ab364d0>





In [ ]:
```
test_path = 'traffic_sign_dataset/Test'
!rm traffic_sign_dataset/Test/GT-final_test.csv
```

In [ ]:
```
from PIL import Image

def scaling(test_images, test_path):
  images = []

  image_path = test_images

  for x in image_path:
    img = Image.open(test_path + '/' + x)
    img = img.resize((50,50))
    img = np.array(img)
    images.append(img)

  images = np.array(images)
  images = images/255

  return images
```

In [ ]:
```
test_images = scaling(sorted(os.listdir(test_path)),test_path)
```

In [ ]:
```
test = pd.read_csv('traffic_sign_dataset/Test.csv')
y_test = test['ClassId'].values
```

In [ ]:
```
# Store the Labels:
all_labels = ['Speed limit (20km/h)','Speed limit (30km/h)','Speed limit (50km/h)','Speed limit (60km/h)',
              'Speed limit (70km/h)','Speed limit (80km/h)','End of speed limit (80km/h)','Speed limit (100km/h)',
              'Speed limit (120km/h)','No passing','No passing for vechiles over 3.5 metric tons',
              'Right-of-way at the next intersection','Priority road','Yield','Stop','No vechiles',
              'Vechiles over 3.5 metric tons prohibited','No entry','General caution','Dangerous curve to the left',
              'Dangerous curve to the right','Double curve','Bumpy road','Slippery road','Road narrows on the right',
              'Road work','Traffic signals','Pedestrians','Children crossing','Bicycles crossing','Beware of ice/snow'
              'Wild animals crossing','End of all speed and passing limits','Turn right ahead','Turn left ahead',
              'Ahead only','Go straight or right','Go straight or left','Keep right','Keep left','Roundabout mandatory
              'End of no passing','End of no passing by vechiles over 3.5 metric']
```

In [ ]:
```
```

In [ ]:
```
# Now the results display shall begin:
img = Image.open(test_path + '/00199.png')
img
```

Out[28]:

In [ ]:
```
#Print the label automatically on its own:
print("Original Label : ",all_labels[y_test[199]])
```

```
Original Label :  Children crossing
```

In [ ]:
```
```