trafic signal

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D,MaxPool2D,Dense,Flatten,Dropout

data=[]
labels=[]
classes=43
cur_path =os.getcwd()

for i in range(classes):
    path = os.path.join(cur_path, "E:\\resume projects\\trafic\\Train", str(i)

    images=os.listdir(path)

    for a in images:
        try:
            image=Image.open(path +'\\'+a)
            image=image.resize((30,30))
            image =np.array(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")
data= np.array(data)
labels =np.array(labels)
```

```
WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python
311\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax
_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_c
ross_entropy instead.
```

In [2]:

```
print(data.shape ,labels.shape)
x_train,x_test,y_train,y_test=train_test_split(data,labels,test_size=0.2,rando

print(x_train.shape ,x_test.shape,y_train.shape,y_test.shape)

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

```
(39209, 30, 30, 3) (39209,)
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

In [3]:

```
model= Sequential()
model.add(Conv2D(filters=32 ,kernel_size=(5,5),activation='relu' ,input_shape=
model.add(Conv2D(filters=32 ,kernel_size=(5,5) ,activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64,kernel_size=(3,3),activation='relu'))
model.add(Conv2D(filters=64,kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43,activation='softmax'))

#compilation of the model
model.compile(loss='categorical_crossentropy' ,optimizer='adam' ,metrics=['acc
```

```
WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python
311\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is
deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python
311\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.
nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python
311\site-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Opt
imizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

step 3: TRAIN AND VALIDATE THE MODEL

In [4]:

```
epochs =15
history =model.fit(x_train,y_train ,batch_size=64,epochs=epochs ,validation_da
Epoch 1/15
WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python
311\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedT
ensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue in
stead.

WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python
311\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.execu
ting_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executi
ng_eagerly_outside_functions instead.

491/491 [==============================] - 20s 37ms/step - loss: 2.5982 - acc
uracy: 0.3817 - val_loss: 0.9276 - val_accuracy: 0.7809
Epoch 2/15
491/491 [==============================] - 18s 36ms/step - loss: 1.0596 - acc
uracy: 0.6798 - val_loss: 0.4332 - val_accuracy: 0.8929
Epoch 3/15
491/491 [==============================] - 18s 37ms/step - loss: 0.7125 - acc
uracy: 0.7846 - val_loss: 0.2991 - val_accuracy: 0.9244
Epoch 4/15
491/491 [==============================] - 18s 37ms/step - loss: 0.5785 - acc
uracy: 0.8245 - val_loss: 0.2185 - val_accuracy: 0.9480
Epoch 5/15
491/491 [==============================] - 19s 40ms/step - loss: 0.4824 - acc
uracy: 0.8542 - val_loss: 0.1961 - val_accuracy: 0.9441
Epoch 6/15
491/491 [==============================] - 18s 37ms/step - loss: 0.4101 - acc
uracy: 0.8742 - val_loss: 0.1755 - val_accuracy: 0.9426
Epoch 7/15
491/491 [==============================] - 18s 36ms/step - loss: 0.3400 - acc
uracy: 0.8960 - val_loss: 0.1373 - val_accuracy: 0.9580
Epoch 8/15
491/491 [==============================] - 18s 36ms/step - loss: 0.3146 - acc
uracy: 0.9037 - val_loss: 0.1328 - val_accuracy: 0.9672
Epoch 9/15
491/491 [==============================] - 18s 36ms/step - loss: 0.2971 - acc
uracy: 0.9117 - val_loss: 0.0874 - val_accuracy: 0.9736
Epoch 10/15
491/491 [==============================] - 18s 36ms/step - loss: 0.2691 - acc
uracy: 0.9188 - val_loss: 0.0932 - val_accuracy: 0.9704
Epoch 11/15
491/491 [==============================] - 19s 38ms/step - loss: 0.2578 - acc
uracy: 0.9235 - val_loss: 0.0966 - val_accuracy: 0.9684
Epoch 12/15
491/491 [==============================] - 23s 47ms/step - loss: 0.2416 - acc
uracy: 0.9273 - val_loss: 0.0973 - val_accuracy: 0.9680
Epoch 13/15
491/491 [==============================] - 23s 47ms/step - loss: 0.2521 - acc
uracy: 0.9245 - val_loss: 0.0678 - val_accuracy: 0.9821
Epoch 14/15
491/491 [==============================] - 23s 47ms/step - loss: 0.2190 - acc
uracy: 0.9350 - val_loss: 0.0627 - val_accuracy: 0.9818
Epoch 15/15
491/491 [==============================] - 23s 47ms/step - loss: 0.2109 - acc
uracy: 0.9364 - val_loss: 0.0647 - val_accuracy: 0.9802
```

In [5]:

```
model.save("my_model.h5")
```

```
C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras
\src\engine\training.py:3103: UserWarning: You are saving your model as an HD
F5 file via `model.save()`. This file format is considered legacy. We recomme
nd using instead the native Keras format, e.g. `model.save('my_model.keras')
`.
  saving_api.save_model(
```

In [6]:

```
#testing accuracy an test dataset

from sklearn.metrics import accuracy_score

y_test = pd.read_csv(r"E:\\resume projects\\trafic\\Test.csv")

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

data=[]
for img in imgs:
    print(img)
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
```

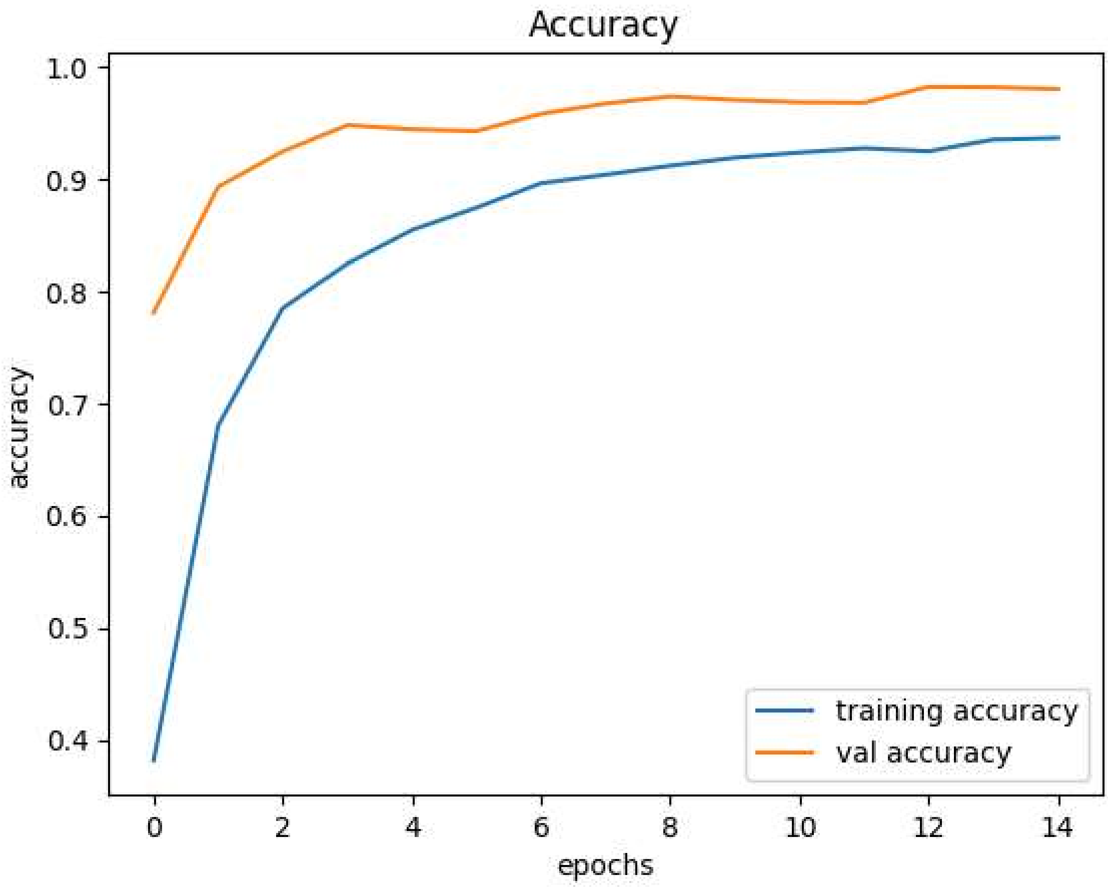
```
Test/00000.png
Test/00001.png
Test/00002.png
Test/00003.png
Test/00004.png
Test/00005.png
Test/00006.png
Test/00007.png
Test/00008.png
Test/00009.png
Test/00010.png
Test/00011.png
Test/00012.png
Test/00013.png
Test/00014.png
Test/00015.png
Test/00016.png
Test/00017.png
Test/00018.png
Test/00019.png
```

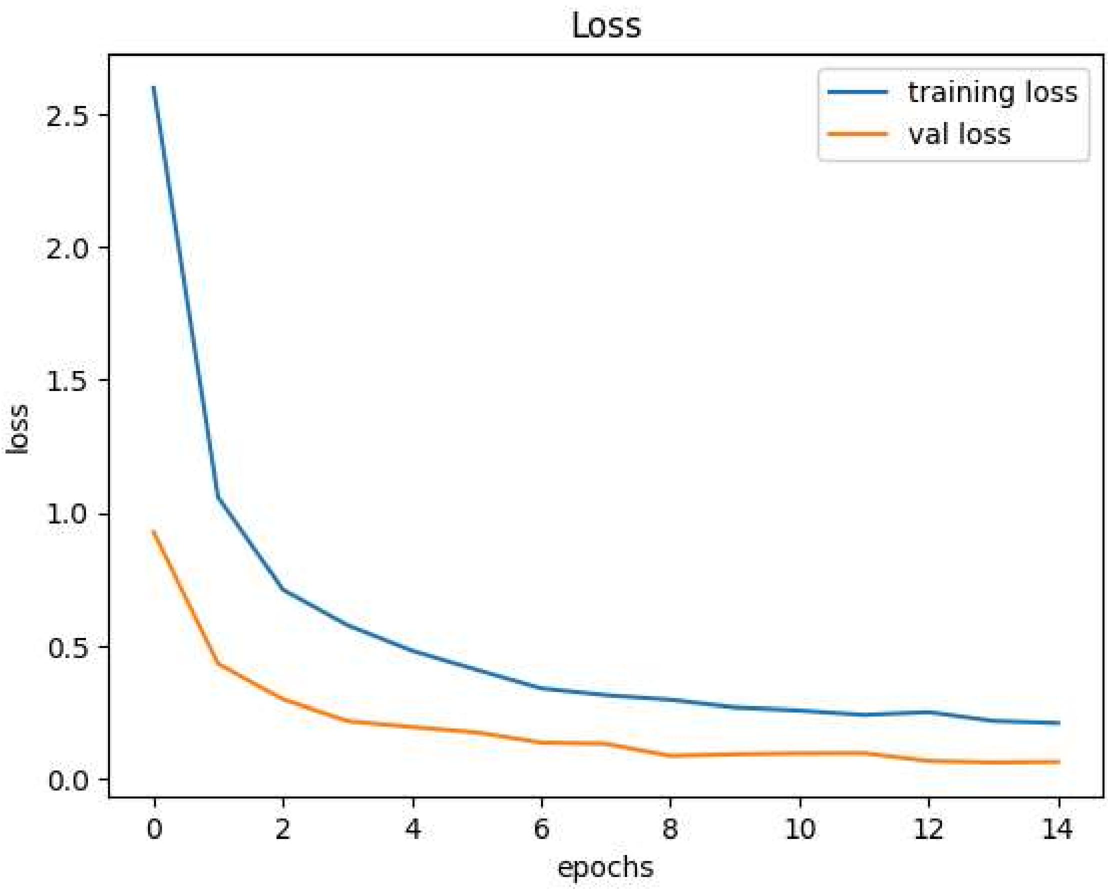
In [7]:

```
plt.figure(0)
plt.plot(history.history['accuracy'],label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()



plt.figure(1)
plt.plot(history.history['loss'] ,label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
```

Out[7]:

```
<matplotlib.legend.Legend at 0x25ac76ad550>
```




Loss
training loss
val loss
loss
2.5
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
12
14
epochs

STEP 4: TEST OUR MODEL WITH TEST DATASET

In [8]:

```python
from sklearn.metrics import accuracy_score
import pandas as pd
from PIL import Image
import numpy as np

# Assuming you have already loaded and compiled your model
# model = ...

y_test = pd.read_csv("E:\\resume projects\\trafic\\Test.csv")

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

data = []

for img in imgs:
    image = Image.open(img)
    image = image.resize((30, 30))
    data.append(np.array(image))

x_test = np.array(data)

# Use predict method to get the probabilities for each class
pred_probabilities = model.predict(x_test)

# Use argmax to get the predicted class
pred = np.argmax(pred_probabilities, axis=1)

# Accuracy with the test data
accuracy = accuracy_score(labels, pred)
print("Accuracy:", accuracy)
```

```
395/395 [==============================] - 5s 11ms/step
Accuracy: 0.9471892319873317
```

#In the end, we are going to save the model that we have trained using the Keras model.save() function.

In [9]:

```python
model.save('traffic_classifier.h5')
```

```
C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras
\src\engine\training.py:3103: UserWarning: You are saving your model as an HD
F5 file via `model.save()`. This file format is considered legacy. We recomme
nd using instead the native Keras format, e.g. `model.save('my_model.keras')
`.
  saving_api.save_model(
```

In [13]:

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image

import numpy
#load the trained model to classify sign
from keras.models import load_model
model = load_model('traffic_classifier.h5')
#dictionary to label all traffic signs class.
classes = { 1:'Speed limit (20km/h)',
            2:'Speed limit (30km/h)',
            3:'Speed limit (50km/h)',
            4:'Speed limit (60km/h)',
            5:'Speed limit (70km/h)',
            6:'Speed limit (80km/h)',
            7:'End of speed limit (80km/h)',
            8:'Speed limit (100km/h)',
            9:'Speed limit (120km/h)',
            10:'No passing',
            11:'No passing veh over 3.5 tons',
            12:'Right-of-way at intersection',
            13:'Priority road',
            14:'Yield',
            15:'Stop',
            16:'No vehicles',
            17:'Veh > 3.5 tons prohibited',
            18:'No entry',
            19:'General caution',
            20:'Dangerous curve left',
            21:'Dangerous curve right',
            22:'Double curve',
            23:'Bumpy road',
            24:'Slippery road',
            25:'Road narrows on the right',
            26:'Road work',
            27:'Traffic signals',
            28:'Pedestrians',
            29:'Children crossing',
            30:'Bicycles crossing',
            31:'Beware of ice/snow',
            32:'Wild animals crossing',
            33:'End speed + passing limits',
            34:'Turn right ahead',
            35:'Turn left ahead',
            36:'Ahead only',
            37:'Go straight or right',
            38:'Go straight or left',
            39:'Keep right',
            40:'Keep left',
            41:'Roundabout mandatory',
            42:'End of no passing',
            43:'End no passing veh > 3.5 tons' }
```

In [14]:

```
def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    pred = model.predict(image)[0]  # Use model.predict instead of predict_gen
    pred_class = np.argmax(pred) + 1  # Get the index of the maximum value
    sign = classes[pred_class]
    print(sign)
    label.configure(foreground='#011638', text=sign)
```

In [15]:

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
#initialise GUI
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')
label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)
def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)

    pred = model.predict([image])
    pred_c = numpy.argmax(pred)+1
    sign = classes[pred_c]
    print(sign)
    label.configure(foreground='#011638', text=sign)
def show_classify_button(file_path):
    classify_b=Button(top,text="Classify Image",command=lambda: classify(file_
    classify_b.configure(background='#364156', foreground='white',font=('arial
    classify_b.place(relx=0.79,rely=0.46)
def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)
        im=ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:

         pass
upload=Button(top,text="Upload an image",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156', foreground='white',font=('arial',10,'bo
upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Know Your Traffic Sign",pady=20, font=('arial',20,'
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

In [ ]: