Driver Behaviour Detction

Problem Statement

Distracted driving, lane departures, and aggressive driving are significant factors contributing to road accidents and fatalities. Traditional methods for monitoring driver behavior often rely on manual observation or basic sensor data, which can be inefficient and less accurate. There is a need for an automated, real-time system to accurately classify and detect various driver behaviors using image data.

Objective

The aim of this project is to develop a robust system that leverages Convolutional Neural Networks (CNNs) to classify driver behavior.

```
import os
import cv2
import numpy as np
```

```
!kaggle datasets download -d robinreni/revitsone-5class
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (server 1.6.15 / client 1.6.14)
Dataset URL: https://www.kaggle.com/datasets/robinreni/revitsone-5class
License(s): DbCL-1.0
revitsone-5class.zip: Skipping, found more recently modified local copy (use --force to force download)
```

```
!unzip /content/revitsone-5class.zip
```

```
Archive:  /content/revitsone-5class.zip
replace Revitsone-5classes/other_activities/2019-04-2416-05-13.png? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

```
x = []
y = []
main_path='/content/Revitsone-5classes'
sub_dirs=os.listdir(main_path)
for subdir in sub_dirs:
    subpath=os.path.join(main_path, subdir)
    img_names=os.listdir(subpath)
    for img_name in img_names:
        img_path=os.path.join(subpath, img_name)
        img_array=cv2.imread(img_path, 0)
        if img_array is not None:
            img_resized=cv2.resize(img_array, (224, 224))
            img_final=img_resized.reshape(224,224,1)
            # img_final=img_final/255.0
            x.append(img_final)
            y.append(subdir)
        else:
            print(f"Skipped empty or unreadable image: {img_name}")
```

```
Skipped empty or unreadable image: img_62337.jpg
Skipped empty or unreadable image: img_8771.jpg
Skipped empty or unreadable image: img_67523.jpg
Skipped empty or unreadable image: img_101434.jpg
Skipped empty or unreadable image: img_84605.jpg
Skipped empty or unreadable image: img_70552.jpg
Skipped empty or unreadable image: img_4664.jpg
Skipped empty or unreadable image: img_13396.jpg
Skipped empty or unreadable image: img_79.jpg
Skipped empty or unreadable image: img_22266.jpg
Skipped empty or unreadable image: img_13625.jpg
Skipped empty or unreadable image: img_13541.jpg
Skipped empty or unreadable image: img_20398.jpg
Skipped empty or unreadable image: img_7973.jpg
Skipped empty or unreadable image: img_13318.jpg
```

```
len(x)
```

```
10751
```

```
len(y)
```

```
10751
```

```
import numpy as np
x_final=np.array(x)
```

```
x_final.shape
```

```
(10751, 224, 224, 1)
```

```
y
```

```
['talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
 'talking_phone',
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y_final=le.fit_transform(y)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_final,y_final,test_size=0.3)
```

```
import tensorflow
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D,MaxPool2D,Flatten,Dense

model=keras.Sequential([Conv2D(32,(3,3),input_shape=(224,224,1),activation='relu'),
                      (MaxPool2D((2,2))),
                       Conv2D(16,(3,3),activation='relu'),
                       MaxPool2D((2,2)),
                       Flatten(),
                       Dense(32,activation='relu'),
                       Dense(16,activation='relu'),
                       Dense(5,activation='softmax')])
```

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

```
acc=model.fit(x_train,y_train,epochs=30, validation_data=(x_test, y_test))
```

```
Epoch 1/30
236/236 [==============================] - 10s 29ms/step - loss: 4.1193 - accuracy: 0.6836 - val_loss: 0.2680 - val_accuracy: 0.
Epoch 2/30
236/236 [==============================] - 6s 27ms/step - loss: 0.1438 - accuracy: 0.9551 - val_loss: 0.1549 - val_accuracy: 0.9
Epoch 3/30
236/236 [==============================] - 6s 27ms/step - loss: 0.0686 - accuracy: 0.9797 - val_loss: 0.1549 - val_accuracy: 0.9
Epoch 4/30
236/236 [==============================] - 6s 27ms/step - loss: 0.0418 - accuracy: 0.9886 - val_loss: 0.1019 - val_accuracy: 0.9
Epoch 5/30
236/236 [==============================] - 6s 26ms/step - loss: 0.0277 - accuracy: 0.9910 - val_loss: 0.1174 - val_accuracy: 0.9
Epoch 6/30
236/236 [==============================] - 7s 28ms/step - loss: 0.0240 - accuracy: 0.9939 - val_loss: 0.1089 - val_accuracy: 0.9
Epoch 7/30
236/236 [==============================] - 6s 26ms/step - loss: 0.0064 - accuracy: 0.9980 - val_loss: 0.1416 - val_accuracy: 0.9
Epoch 8/30
236/236 [==============================] - 6s 27ms/step - loss: 0.0337 - accuracy: 0.9915 - val_loss: 0.1226 - val_accuracy: 0.9
Epoch 9/30
236/236 [==============================] - 6s 26ms/step - loss: 0.0289 - accuracy: 0.9915 - val_loss: 0.1429 - val_accuracy: 0.9
Epoch 10/30
236/236 [==============================] - 6s 27ms/step - loss: 0.0288 - accuracy: 0.9922 - val_loss: 0.1432 - val_accuracy: 0.9
Epoch 11/30
236/236 [==============================] - 6s 26ms/step - loss: 0.0293 - accuracy: 0.9915 - val_loss: 0.1900 - val_accuracy: 0.9
Epoch 12/30
236/236 [==============================] - 6s 27ms/step - loss: 0.0365 - accuracy: 0.9910 - val_loss: 0.1950 - val_accuracy: 0.9
Epoch 13/30
236/236 [==============================] - 6s 26ms/step - loss: 0.0402 - accuracy: 0.9908 - val_loss: 0.1616 - val_accuracy: 0.9
Epoch 14/30
236/236 [==============================] - 6s 27ms/step - loss: 0.0286 - accuracy: 0.9919 - val_loss: 0.1167 - val_accuracy: 0.9
Epoch 15/30
236/236 [==============================] - 6s 26ms/step - loss: 0.0064 - accuracy: 0.9984 - val_loss: 0.1219 - val_accuracy: 0.9
Epoch 16/30
236/236 [==============================] - 6s 27ms/step - loss: 0.0038 - accuracy: 0.9989 - val_loss: 0.1370 - val_accuracy: 0.9
Epoch 17/30
236/236 [==============================] - 7s 29ms/step - loss: 0.0212 - accuracy: 0.9943 - val_loss: 0.2481 - val_accuracy: 0.9
Epoch 18/30
236/236 [==============================] - 7s 28ms/step - loss: 0.0586 - accuracy: 0.9841 - val_loss: 0.1514 - val_accuracy: 0.9
Epoch 19/30
236/236 [==============================] - 6s 27ms/step - loss: 0.0190 - accuracy: 0.9956 - val_loss: 0.1722 - val_accuracy: 0.9
Epoch 20/30
236/236 [==============================] - 7s 30ms/step - loss: 0.0134 - accuracy: 0.9967 - val_loss: 0.1444 - val_accuracy: 0.9
Epoch 21/30
236/236 [==============================] - 6s 26ms/step - loss: 0.0053 - accuracy: 0.9983 - val_loss: 0.1270 - val_accuracy: 0.9
Epoch 22/30
236/236 [==============================] - 6s 28ms/step - loss: 6.2634e-04 - accuracy: 1.0000 - val_loss: 0.1221 - val_accuracy:
Epoch 23/30
236/236 [==============================] - 6s 26ms/step - loss: 2.1294e-04 - accuracy: 1.0000 - val_loss: 0.1245 - val_accuracy:
Epoch 24/30
236/236 [==============================] - 7s 30ms/step - loss: 1.2513e-04 - accuracy: 1.0000 - val_loss: 0.1264 - val_accuracy:
Epoch 25/30
236/236 [==============================] - 8s 36ms/step - loss: 9.3942e-05 - accuracy: 1.0000 - val_loss: 0.1274 - val_accuracy:
Epoch 26/30
236/236 [==============================] - 8s 32ms/step - loss: 7.4383e-05 - accuracy: 1.0000 - val_loss: 0.1290 - val_accuracy:
Epoch 27/30
236/236 [==============================] - 6s 27ms/step - loss: 5.2033e-05 - accuracy: 1.0000 - val_loss: 0.1322 - val_accuracy:
Epoch 28/30
236/236 [==============================] - 6s 27ms/step - loss: 4.1204e-05 - accuracy: 1.0000 - val_loss: 0.1333 - val_accuracy:
Epoch 29/30
```

```
model.evaluate(x_test,y_test)
```

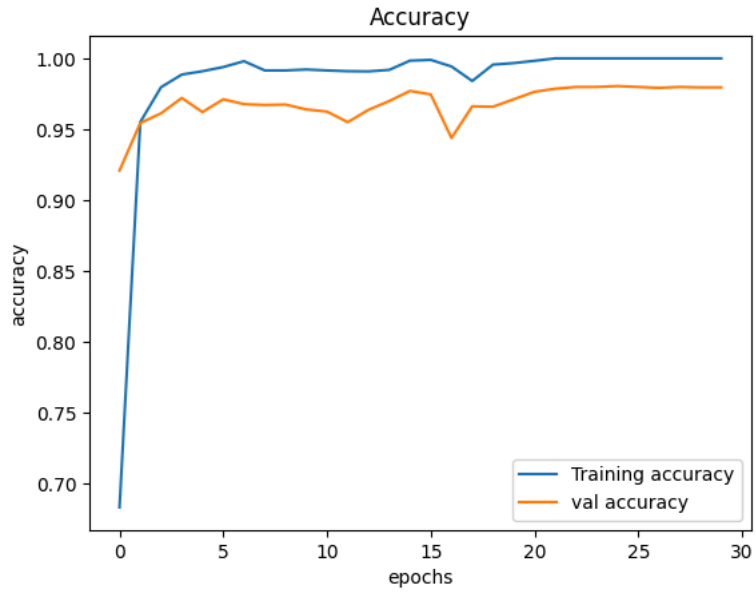
```
101/101 [==============================] - 1s 9ms/step - loss: 0.1374 - accuracy: 0.9795
[0.13744047284126282, 0.9795412421226501]
```

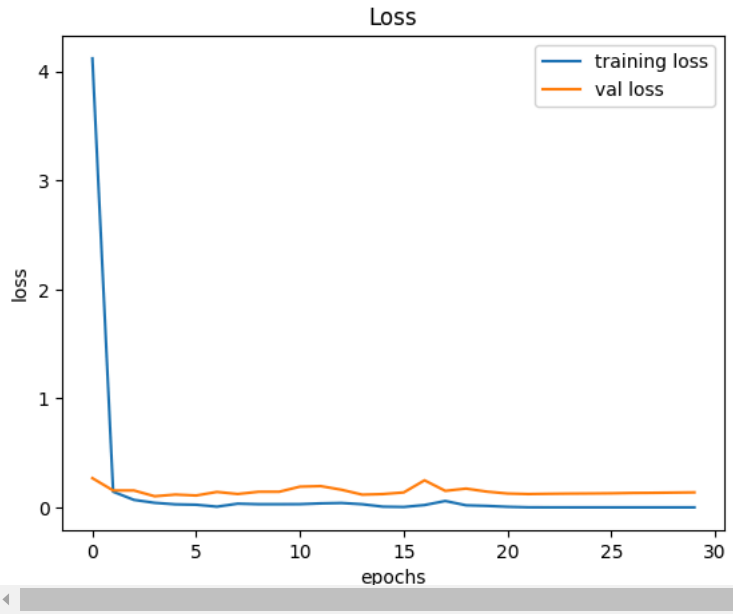
```
def prediction(path):
  array=cv2.imread(path,0)
  arr_resize=cv2.resize(array,(224,224))
  arr_reshape=arr_resize.reshape(1,224,224,1)
  arr_reshape=arr_reshape/255
  y_pred=model.predict(arr_reshape)
  pred_label=np.argmax(y_pred)
  pred_label=le.inverse_transform([pred_label])
  return pred_label
```

```
prediction('/content/Turning_while_driving.jpg')
```

```
1/1 [==============================] - 0s 20ms/step
array(['turning'], dtype='<U16')
```

```
#plotting graphs for accuracy
import matplotlib.pyplot as plt
plt.figure(0)
plt.plot(acc.history['accuracy'],label='Training accuracy')
plt.plot(acc.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
plt.figure(1)
plt.plot(acc.history['loss'], label='training loss')
plt.plot(acc.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```





Conclusion

This project aims to enhance road safety by using CNNs to accurately detect and classify driver behaviors in real-time, achieving a validation accuracy of 0.9795