The project given in here is to observe the images of a number of people driving a car and detect the category they fall into. This is done with the aim to observe the activity of drivers and warn them whenever they arre distracted so as to avoid street accidents.

In [1]:

```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/state-farm-distracted-driver-detection/sample_submission.csv
/kaggle/input/state-farm-distracted-driver-detection/driver_imgs_list.csv
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_12848.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_26944.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_51989.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_50852.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_31481.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_46523.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_16245.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_8456.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_24370.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_56073.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_18964.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_35710.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_12011.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_64192.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_98904.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_33126.jpg
/kaggle/input/state-farm-distracted-driver-detection/imgs/test/img_30683.jpg
```

In [2]:

```
import tensorflow
os.environ['KERAS_BACKEND'] = 'tensorflow'
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # 3 = INFO, WARNING, and ERROR messages are not printed

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.preprocessing.image import ImageDataGenerator
```

In [3]:

```
dataset = pd.read_csv('../input/state-farm-distracted-driver-detection/driver_imgs_list.csv')
dataset.head(5)
```

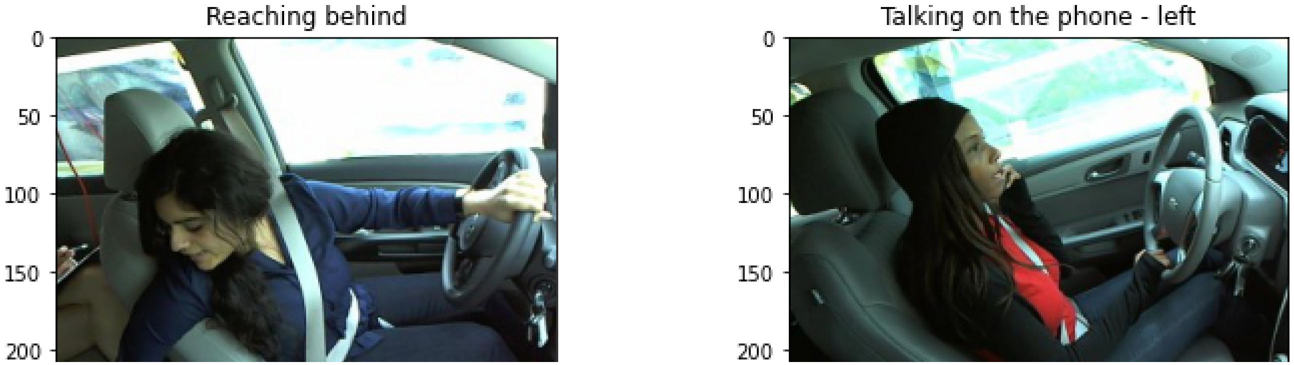
Out[3]:

	subject	classname	img
0	p012	c0	img_10206.jpg
1	p012	c0	img_27079.jpg
2	p012	c0	img_50749.jpg
3	p012	c0	img_97089.jpg
4	p012	c0	img_37741.jpg

In [4]:
```
import os
from IPython.display import display, Image
import matplotlib.image as mpimg

activity_map = {'c0': 'Safe driving',
                'c1': 'Texting - right',
                'c2': 'Talking on the phone - right',
                'c3': 'Texting - left',
                'c4': 'Talking on the phone - left',
                'c5': 'Operating the radio',
                'c6': 'Drinking',
                'c7': 'Reaching behind',
                'c8': 'Hair and makeup',
                'c9': 'Talking to passenger'}

plt.figure(figsize = (12, 20))
image_count = 1
BASE_URL = '../input/state-farm-distracted-driver-detection/imgs/train/'
for directory in os.listdir(BASE_URL):
    if directory[0] != '.':
        for i, file in enumerate(os.listdir(BASE_URL + directory)):
            if i == 1:
                break
            else:
                fig = plt.subplot(5, 2, image_count)
                image_count += 1
                image = mpimg.imread(BASE_URL + directory + '/' + file)
                plt.imshow(image)
                plt.title(activity_map[directory])
```


Reaching behind

Talking on the phone - left

Texting - right

Drinking

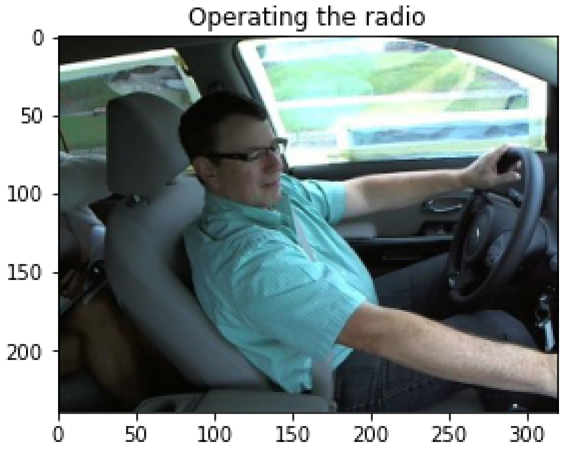
Talking on the phone - right

Talking to passenger

Hair and makeup

Safe driving

Texting - left


Operating the radio

Creating the model

In [16]:

```
classifier = Sequential()
classifier.add(Conv2D(filters = 128, kernel_size = (3, 3), activation = 'relu', input_shape = (240, 240, 3), data_format
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Conv2D(filters = 32, kernel_size = (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Flatten())
classifier.add(Dense(units = 1024, activation = 'relu'))
classifier.add(Dense(units = 256, activation = 'relu'))
classifier.add(Dense(units = 10, activation = 'softmax'))
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
classifier.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 238, 238, 128)     3584      
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 119, 119, 128)     0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 117, 117, 64)      73792     
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 58, 58, 64)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 32)        18464     
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 28, 28, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              25691136  
_________________________________________________________________
dense_4 (Dense)              (None, 256)               262400    
_________________________________________________________________
dense_5 (Dense)              (None, 10)                2570      
=================================================================
Total params: 26,051,946
Trainable params: 26,051,946
```

```
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

```
train_datagen = ImageDataGenerator(rescale = 1.0/255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   validation_split = 0.2)

training_set = train_datagen.flow_from_directory('../input/state-farm-distracted-driver-detection/imgs/train',
                                                 target_size = (240, 240),
                                                 batch_size = 32,
                                                 subset = 'training')

validation_set = train_datagen.flow_from_directory('../input/state-farm-distracted-driver-detection/imgs/train',
                                                   target_size = (240, 240),
                                                   batch_size = 32,
                                                   subset = 'validation')
```

```
Found 13975 images belonging to 10 classes.
Found 3487 images belonging to 10 classes.
```

In [19]:

```
classifier.fit_generator(training_set,
                         epochs = 5,
                         validation_data = validation_set)
```

```
Epoch 1/5
437/437 [==============================] - 276s 633ms/step - loss: 1.2645 - accuracy: 0.5516 - val_loss: 0.5609 - val_a
ccuracy: 0.8216
Epoch 2/5
437/437 [==============================] - 276s 631ms/step - loss: 0.3649 - accuracy: 0.8829 - val_loss: 0.3172 - val_a
ccuracy: 0.8988
Epoch 3/5
437/437 [==============================] - 274s 626ms/step - loss: 0.2253 - accuracy: 0.9311 - val_loss: 0.1987 - val_a
ccuracy: 0.9429
Epoch 4/5
437/437 [==============================] - 273s 624ms/step - loss: 0.1698 - accuracy: 0.9486 - val_loss: 0.1638 - val_a
ccuracy: 0.9495
Epoch 5/5
437/437 [==============================] - 271s 620ms/step - loss: 0.1306 - accuracy: 0.9579 - val_loss: 0.1888 - val_a
ccuracy: 0.9429
```

Out[19]:

```
<tensorflow.python.keras.callbacks.History at 0x7fa0a0557090>
```

So the training accuracy comes out to be 96% and the testing accuracy is 94%

In [20]:

```
classifier.save("state_farm_distracted_driver_detection_model3.h5")
```

In [21]:

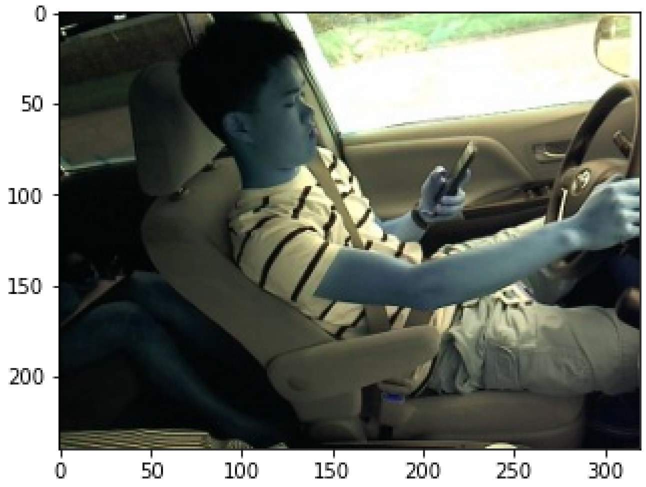
```
classes_predict= ['safe driving',
'texting - right',
'talking on the phone - right',
'texting - left',
'talking on the phone - left',
'operating the radio',
'drinking',
'reaching behind',
'hair and makeup',
'talking to passenger']
```

Testing on an image from test data

In [24]:
```
import cv2
frame= cv2.imread('../input/state-farm-distracted-driver-detection/imgs/test/img_10034.jpg')
frame2= cv2.resize(frame, (240,240))
img_cv_predict = np.reshape(frame2, [1,240,240,3])
arr_predict =classifier.predict(img_cv_predict, batch_size =1)
print(arr_predict)
print(classes_predict[np.argmax(arr_predict)])
plt.imshow(frame)
```

```
[[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
texting - left
```

Out[24]:
```
<matplotlib.image.AxesImage at 0x7fa46cff0b50>
```



So we see that the model correctly classifies the image

In [ ]: