```
import pickle

#unpacking the file
with open('/content/drive/MyDrive/Colab Notebooks/Distracted Driver Detection/images.p',
    images = pickle.load(f)

with open('/content/drive/MyDrive/Colab Notebooks/Distracted Driver Detection/labels.p',
    labels = pickle.load(f)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
print(images.shape)
print(labels.shape)
```

```
(22424, 100, 100)
(22424,)
```

```
set(labels)
```

```
{'c0', 'c1', 'c2', 'c3', 'c4', 'c5', 'c6', 'c7', 'c8', 'c9'}
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
labels = le.fit_transform(labels)
```

```
set(labels)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

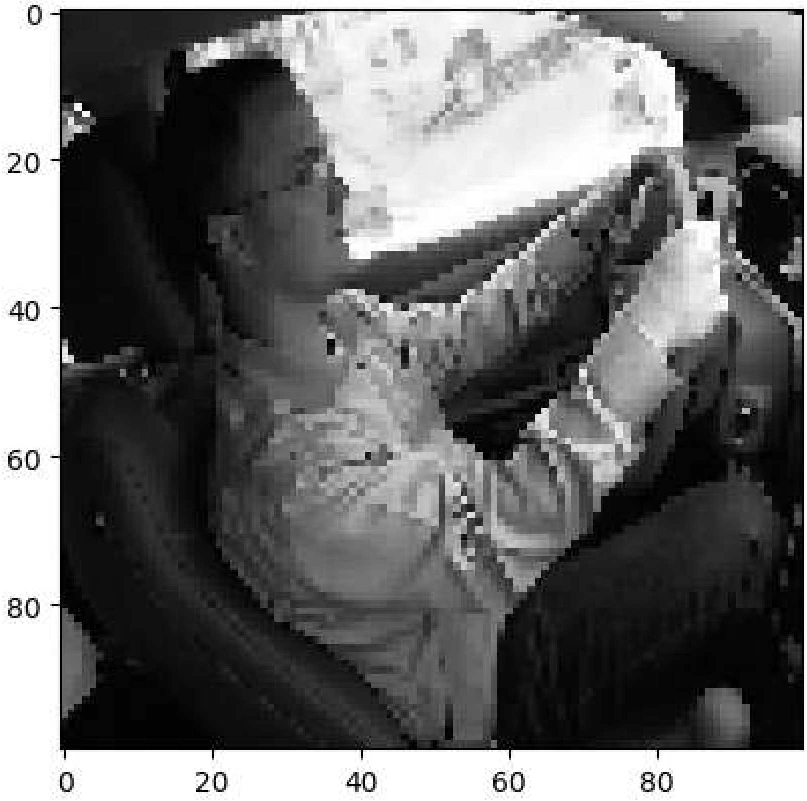
```
import numpy as np
```

```
n_persons = len(set(labels))
print("Number of persons: ", n_persons)
label_mapping = le.inverse_transform(np.arange(n_persons))
for i in range(len(label_mapping)):
  print(i, "-->", label_mapping[i])
```

```
Number of persons:  10
0 --> c0
1 --> c1
2 --> c2
3 --> c3
4 --> c4
5 --> c5
6 --> c6
```

```
     7 --> c7
     8 --> c8
     9 --> c9
```

```
import matplotlib.pyplot as plt

plt.imshow(images[77], cmap=plt.get_cmap("gray"))
plt.show()
```



```
import cv2
```

```
def preprocessing(img):
  img = cv2.equalizeHist(img)
  img = img.reshape(100, 100, 1)
  img = img/255
  return img
```

```
images = np.array(list(map(preprocessing, images)))
print("Shape of Input: ", images.shape)
```

```
Shape of Input:  (22424, 100, 100, 1)
```

```
from tensorflow.keras.utils import to_categorical
```

```
labels = to_categorical(labels)
```

```
categories = labels.shape[1]
print(categories)
```

```
10
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import cifar10

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
# import convolution layer
from tensorflow.keras.layers import Conv2D
# import pooling layer
from tensorflow.keras.layers import MaxPooling2D
# import faltten layer
from tensorflow.keras.layers import Flatten
from tensorflow.keras.optimizers import RMSprop

from tensorflow.keras.layers import Input, Add, Dense, Activation, ZeroPadding2D,GlobalAv

from keras.layers import Conv2D, MaxPooling2D, BatchNormalization

from keras.callbacks import ReduceLROnPlateau


model = Sequential()

model.add(Conv2D(32, (3, 3), padding="same",input_shape = (100,100,1)))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=1))
model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=1))

model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=1))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=1))

model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=1))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dense(10))
model.add(Activation("softmax"))
#model.build((0,100,100,1))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:1
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# Print the model summary
model.summary()
```

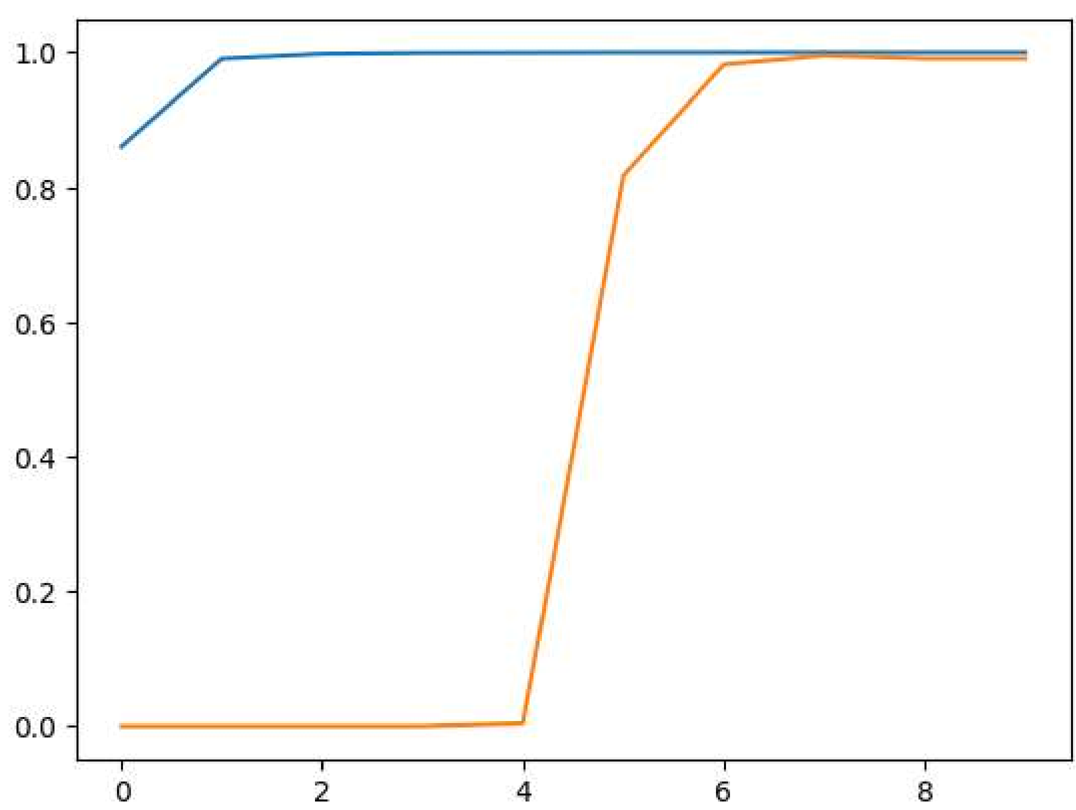
Model: "sequential"

Layer (type)	Output Shape	P
conv2d (Conv2D)	(None, 100, 100, 32)	
activation (Activation)	(None, 100, 100, 32)	
batch_normalization (BatchNormalization)	(None, 100, 100, 32)	
max_pooling2d (MaxPooling2D)	(None, 33, 33, 32)	
conv2d_1 (Conv2D)	(None, 33, 33, 64)	
activation_1 (Activation)	(None, 33, 33, 64)	
batch_normalization_1 (BatchNormalization)	(None, 33, 33, 64)	
conv2d_2 (Conv2D)	(None, 33, 33, 64)	
activation_2 (Activation)	(None, 33, 33, 64)	
batch_normalization_2 (BatchNormalization)	(None, 33, 33, 64)	
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 64)	
conv2d_3 (Conv2D)	(None, 16, 16, 128)	
activation_3 (Activation)	(None, 16, 16, 128)	
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 128)	
conv2d_4 (Conv2D)	(None, 16, 16, 128)	1
activation_4 (Activation)	(None, 16, 16, 128)	
batch_normalization_4 (BatchNormalization)	(None, 16, 16, 128)	
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 128)	
flatten (Flatten)	(None, 8192)	
dense (Dense)	(None, 1024)	8,
activation_5 (Activation)	(None, 1024)	
batch_normalization_5 (BatchNormalization)	(None, 1024)	
dense_1 (Dense)	(None, 10)	
activation_6 (Activation)	(None, 10)	

```
learning_rate_reduction = ReduceLROnPlateau(monitor='accuracy',
                                            patience = 2,
                                            verbose=1,
                                            factor=0.1,
                                            min_lr=0.000001)

opt = tf.keras.optimizers.Adam(learning_rate=0.0001)


#compiling the model
model.compile(RMSprop(learning_rate=0.0001), loss="categorical_crossentropy", metrics=['a


h = model.fit(images,labels,validation_split=0.01,batch_size=250,epochs=10,verbose=1)
```

```
Epoch 1/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 28s 177ms/step - accuracy: 0.6929 - loss: 1.0358 - val_acc
Epoch 2/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9890 - loss: 0.0559 - val_accur
Epoch 3/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9983 - loss: 0.0135 - val_accur
Epoch 4/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 10s 79ms/step - accuracy: 0.9985 - loss: 0.0095 - val_accu
Epoch 5/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9994 - loss: 0.0052 - val_accur
Epoch 6/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 10s 79ms/step - accuracy: 1.0000 - loss: 6.2506e-04 - val_
Epoch 7/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 10s 79ms/step - accuracy: 1.0000 - loss: 3.6429e-04 - val_
Epoch 8/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 10s 80ms/step - accuracy: 1.0000 - loss: 2.7527e-04 - val_
Epoch 9/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 7s 82ms/step - accuracy: 1.0000 - loss: 2.1849e-04 - val_a
Epoch 10/10
89/89 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 1.0000 - loss: 1.8575e-04 - val_a
```

```
plt.plot(h.history['accuracy'])
plt.plot(h.history['val_accuracy'])
plt.show()
```



```
from tensorflow.keras.models import Model
```

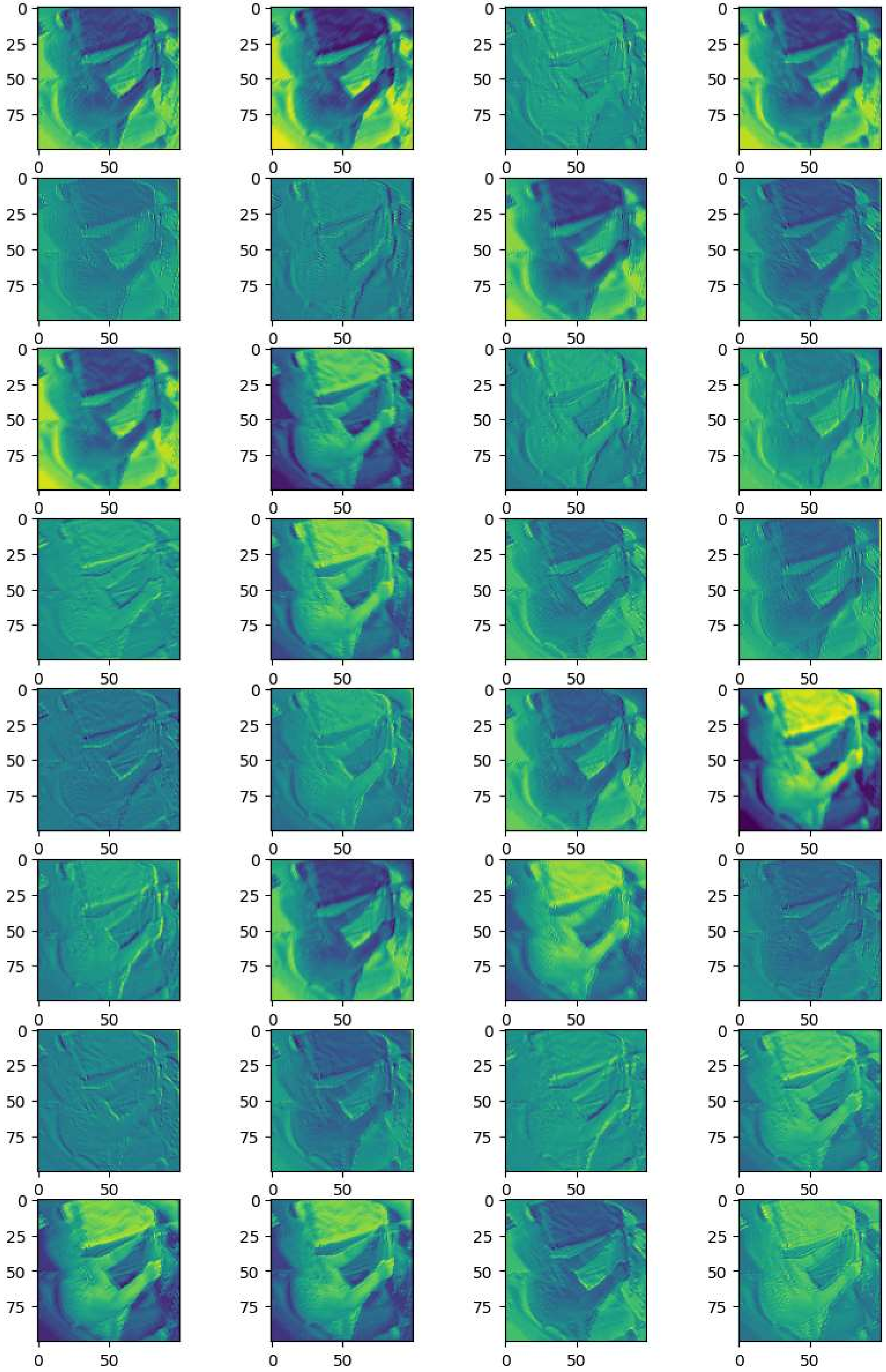
```
layer0 = Model(model.layers[0].input, model.layers[0].output)
features = layer0.predict(images[69].reshape(1,100,100,1))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 158ms/step
```

```
features.shape
```

```
(1, 100, 100, 32)
```

```
plt.figure(figsize=(10,15))
for i in range(32):
  axes = plt.subplot(8, 4, i+1)
  plt.imshow(features[0,:,:,i])
```


0
25
50
75
0
50

```
from google.colab import files
upload=files.upload()
```

Choose Files img_10002.jpg

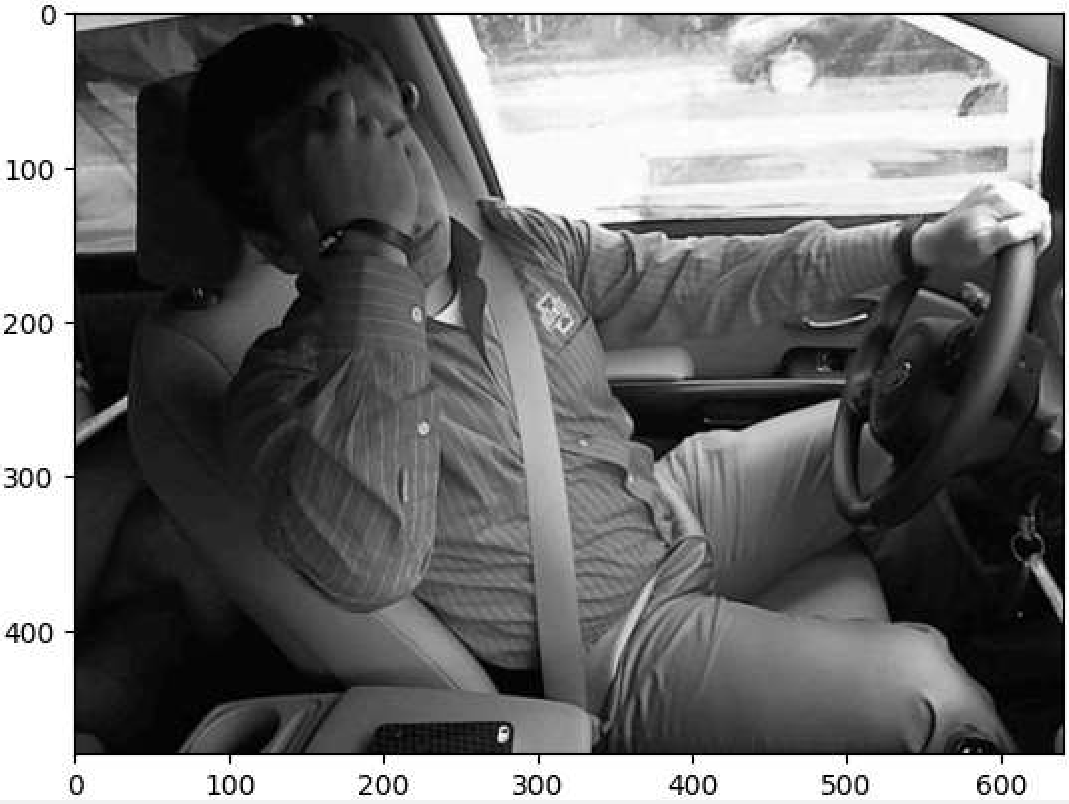
- **img_10002.jpg**(image/jpeg) - 45103 bytes, last modified: 12/16/2019 - 100% done

```
Saving img_10002.jpg to img_10002.jpg
```

```
d=list(upload.keys())[0]
```

```
import cv2
a=np.fromstring(upload[d],np.uint8)
img=cv2.imdecode(a,cv2.IMREAD_COLOR)
img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
print(a)
plt.imshow(img,cmap=plt.get_cmap("gray"))
```

```
[255 216 255 ...  67 255 217]
<ipython-input-32-6cc6288e281e>:2: DeprecationWarning: The binary mode of fromstring
  a=np.fromstring(upload[d],np.uint8)
<matplotlib.image.AxesImage at 0x7fddec3cd720>
```



```
img=np.asarray(img)
img=cv2.resize(img,(100,100))
img=preprocessing(img)


img=img.reshape(1,100,100,1)
print(model.predict(img))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 886ms/step
[[0.00090109 0.07547089 0.00073166 0.13058883 0.25408995 0.00786136
  0.00757964 0.12722619 0.36082283 0.03472747]]
```

```
prediction=model.predict(img)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```

```
p=np.argmax(prediction,axis=1)
l = p.tolist()
l
```

```
[8]
```

```
for i in l:
  if i == 0:
    print('Safe driving')
  elif i == 1:
    print('Texting - right')
  elif i == 2:
    print('Talking on the phone - right')
  elif i == 3:
```