O

m o m $_{s}et : Nx_{t}mpa_{t}l_{t}rim_{s}paces : n1_{s}et : Nx_{t}mpb_{t}l_{r}ange : Nnn_{t}mpa_{t}l28_{i}f_{e}q :$
$VnT_{t}mpb_{t}lchapter_{s}et : Nx_{t}mpa_{t}l_{i}tem : Nn_{t}mpa_{t}l-1_{i}f_{e}q : VnTF_{t}mpa_{t}l*1433141[3]4$

DriverdistractionML

January 11, 2022

1 Driver distraction using Convolutional Neural Networks

1.0.1 Libraries

[1]:
```
import os
from os.path import join
import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import pandas as pd
```

1.0.2 Model

[2]:
```
cnnmodel = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(100, 100,␣
 ↪3)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding = 'same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', padding = 'same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu', padding = 'same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
    ])
cnnmodel.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics␣
 ↪= ['accuracy'])
cnnmodel.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 32)        896
_________________________________________________________________
batch_normalization (BatchNo (None, 98, 98, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 49, 49, 64)        18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 49, 49, 64)        256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 49, 49, 64)        36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 49, 49, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
flatten (Flatten)            (None, 73728)             0
_________________________________________________________________
dropout (Dropout)            (None, 73728)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              75498496
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 76,159,754
Trainable params: 76,159,178
Non-trainable params: 576
_________________________________________________________________
```

1.0.3 Data preprocessing

[3]:
```
workingdir = os.path.abspath('')
trainingdirectory = os.path.join(workingdir + '/
 ↪state-farm-distracted-driver-detection/imgs/train/')
```

1.0.4 Train and validation dataset split

[4]:
```
trainingdataimage = ImageDataGenerator(rescale = 1./255, rotation_range = 40,␣
↪width_shift_range = 0.2,
                                       height_shift_range = 0.2, shear_range =␣
↪0.2, zoom_range = 0.2,
                                       horizontal_flip = True, fill_mode =␣
↪'nearest', validation_split = 0.2)
trainingset = trainingdataimage.flow_from_directory(trainingdirectory,
                                                    target_size = (100,␣
↪100), batch_size = 64,
                                                    class_mode =␣
↪'categorical', subset = 'training', shuffle = True)
validationset = trainingdataimage.flow_from_directory(trainingdirectory,
                                                    target_size = (100,␣
↪100), batch_size = 64,
                                                    class_mode =␣
↪'categorical', subset = 'validation', shuffle = True)
```

```
Found 17943 images belonging to 10 classes.
Found 4481 images belonging to 10 classes.
```

1.0.5 Model fit

[5]:
```
history =  cnnmodel.fit(trainingset, epochs = 60, steps_per_epoch =␣
↪len(trainingset),
                            validation_data = validationset, verbose = 1,␣
↪validation_steps = len(validationset))
```

```
Epoch 1/60
281/281 [==============================] - 114s 391ms/step - loss: 2.3757 -
accuracy: 0.1876 - val_loss: 9.6513 - val_accuracy: 0.1033
Epoch 2/60
281/281 [==============================] - 110s 389ms/step - loss: 1.7734 -
accuracy: 0.3377 - val_loss: 2.3622 - val_accuracy: 0.2540
Epoch 3/60
281/281 [==============================] - 110s 391ms/step - loss: 1.4419 -
accuracy: 0.4602 - val_loss: 1.5239 - val_accuracy: 0.4300
Epoch 4/60
281/281 [==============================] - 109s 389ms/step - loss: 1.1783 -
accuracy: 0.5696 - val_loss: 1.1394 - val_accuracy: 0.5876
Epoch 5/60
281/281 [==============================] - 110s 390ms/step - loss: 0.9663 -
accuracy: 0.6582 - val_loss: 1.4248 - val_accuracy: 0.5242
Epoch 6/60
281/281 [==============================] - 110s 392ms/step - loss: 0.7862 -
accuracy: 0.7288 - val_loss: 0.9391 - val_accuracy: 0.6800
```

```
Epoch 7/60
281/281 [==============================] - 110s 390ms/step - loss: 0.6675 -
accuracy: 0.7702 - val_loss: 1.0366 - val_accuracy: 0.6637
Epoch 8/60
281/281 [==============================] - 110s 392ms/step - loss: 0.5899 -
accuracy: 0.8018 - val_loss: 0.7342 - val_accuracy: 0.7581
Epoch 9/60
281/281 [==============================] - 110s 391ms/step - loss: 0.5231 -
accuracy: 0.8212 - val_loss: 0.9168 - val_accuracy: 0.7304
Epoch 10/60
281/281 [==============================] - 110s 390ms/step - loss: 0.4707 -
accuracy: 0.8436 - val_loss: 0.9220 - val_accuracy: 0.7364
Epoch 11/60
281/281 [==============================] - 110s 391ms/step - loss: 0.4250 -
accuracy: 0.8570 - val_loss: 0.6162 - val_accuracy: 0.8045
Epoch 12/60
281/281 [==============================] - 110s 391ms/step - loss: 0.4047 -
accuracy: 0.8667 - val_loss: 0.8444 - val_accuracy: 0.7193
Epoch 13/60
281/281 [==============================] - 110s 391ms/step - loss: 0.3838 -
accuracy: 0.8721 - val_loss: 0.7112 - val_accuracy: 0.7706
Epoch 14/60
281/281 [==============================] - 110s 393ms/step - loss: 0.3554 -
accuracy: 0.8835 - val_loss: 0.3851 - val_accuracy: 0.8764
Epoch 15/60
281/281 [==============================] - 111s 393ms/step - loss: 0.3435 -
accuracy: 0.8886 - val_loss: 0.3924 - val_accuracy: 0.8744
Epoch 16/60
281/281 [==============================] - 111s 392ms/step - loss: 0.3276 -
accuracy: 0.8963 - val_loss: 0.5650 - val_accuracy: 0.8201
Epoch 17/60
281/281 [==============================] - 110s 390ms/step - loss: 0.3142 -
accuracy: 0.8964 - val_loss: 0.4193 - val_accuracy: 0.8592
Epoch 18/60
281/281 [==============================] - 126s 449ms/step - loss: 0.3005 -
accuracy: 0.9049 - val_loss: 0.9475 - val_accuracy: 0.7242
Epoch 19/60
281/281 [==============================] - 113s 401ms/step - loss: 0.2899 -
accuracy: 0.9072 - val_loss: 0.4327 - val_accuracy: 0.8596
Epoch 20/60
281/281 [==============================] - 111s 395ms/step - loss: 0.2800 -
accuracy: 0.9116 - val_loss: 0.4845 - val_accuracy: 0.8453
Epoch 21/60
281/281 [==============================] - 112s 397ms/step - loss: 0.2797 -
accuracy: 0.9104 - val_loss: 0.3755 - val_accuracy: 0.8842
Epoch 22/60
281/281 [==============================] - 110s 392ms/step - loss: 0.2530 -
accuracy: 0.9185 - val_loss: 1.1566 - val_accuracy: 0.6987
```

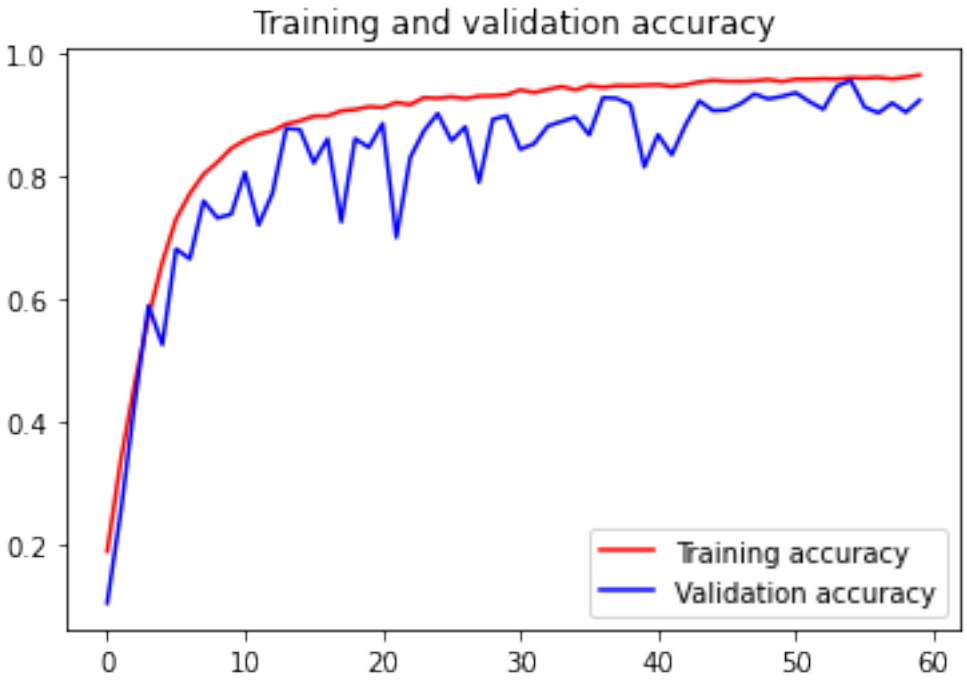
```
Epoch 23/60
281/281 [==============================] - 111s 394ms/step - loss: 0.2626 -
accuracy: 0.9151 - val_loss: 0.6235 - val_accuracy: 0.8286
Epoch 24/60
281/281 [==============================] - 110s 392ms/step - loss: 0.2279 -
accuracy: 0.9265 - val_loss: 0.4207 - val_accuracy: 0.8723
Epoch 25/60
281/281 [==============================] - 111s 394ms/step - loss: 0.2392 -
accuracy: 0.9253 - val_loss: 0.3143 - val_accuracy: 0.9005
Epoch 26/60
281/281 [==============================] - 111s 393ms/step - loss: 0.2376 -
accuracy: 0.9278 - val_loss: 0.5371 - val_accuracy: 0.8561
Epoch 27/60
281/281 [==============================] - 111s 394ms/step - loss: 0.2296 -
accuracy: 0.9249 - val_loss: 0.4364 - val_accuracy: 0.8788
Epoch 28/60
281/281 [==============================] - 112s 399ms/step - loss: 0.2225 -
accuracy: 0.9289 - val_loss: 0.7858 - val_accuracy: 0.7884
Epoch 29/60
281/281 [==============================] - 112s 397ms/step - loss: 0.2192 -
accuracy: 0.9294 - val_loss: 0.3663 - val_accuracy: 0.8909
Epoch 30/60
281/281 [==============================] - 111s 395ms/step - loss: 0.2199 -
accuracy: 0.9308 - val_loss: 0.3474 - val_accuracy: 0.8971
Epoch 31/60
281/281 [==============================] - 111s 393ms/step - loss: 0.1921 -
accuracy: 0.9390 - val_loss: 0.5469 - val_accuracy: 0.8422
Epoch 32/60
281/281 [==============================] - 111s 393ms/step - loss: 0.2081 -
accuracy: 0.9347 - val_loss: 0.5347 - val_accuracy: 0.8514
Epoch 33/60
281/281 [==============================] - 111s 394ms/step - loss: 0.1980 -
accuracy: 0.9399 - val_loss: 0.3975 - val_accuracy: 0.8802
Epoch 34/60
281/281 [==============================] - 111s 393ms/step - loss: 0.1897 -
accuracy: 0.9445 - val_loss: 0.3922 - val_accuracy: 0.8875
Epoch 35/60
281/281 [==============================] - 111s 393ms/step - loss: 0.1963 -
accuracy: 0.9390 - val_loss: 0.3592 - val_accuracy: 0.8947
Epoch 36/60
281/281 [==============================] - 111s 394ms/step - loss: 0.1797 -
accuracy: 0.9461 - val_loss: 0.4619 - val_accuracy: 0.8657
Epoch 37/60
281/281 [==============================] - 110s 390ms/step - loss: 0.1861 -
accuracy: 0.9432 - val_loss: 0.2377 - val_accuracy: 0.9264
Epoch 38/60
281/281 [==============================] - 111s 395ms/step - loss: 0.1728 -
accuracy: 0.9461 - val_loss: 0.2581 - val_accuracy: 0.9252
```

```
Epoch 39/60
281/281 [==============================] - 111s 396ms/step - loss: 0.1836 -
accuracy: 0.9458 - val_loss: 0.2636 - val_accuracy: 0.9154
Epoch 40/60
281/281 [==============================] - 110s 392ms/step - loss: 0.1711 -
accuracy: 0.9468 - val_loss: 0.6902 - val_accuracy: 0.8137
Epoch 41/60
281/281 [==============================] - 111s 394ms/step - loss: 0.1743 -
accuracy: 0.9476 - val_loss: 0.4437 - val_accuracy: 0.8661
Epoch 42/60
281/281 [==============================] - 111s 393ms/step - loss: 0.1788 -
accuracy: 0.9447 - val_loss: 0.5826 - val_accuracy: 0.8333
Epoch 43/60
281/281 [==============================] - 110s 391ms/step - loss: 0.1790 -
accuracy: 0.9473 - val_loss: 0.3731 - val_accuracy: 0.8822
Epoch 44/60
281/281 [==============================] - 111s 395ms/step - loss: 0.1613 -
accuracy: 0.9521 - val_loss: 0.2974 - val_accuracy: 0.9210
Epoch 45/60
281/281 [==============================] - 113s 403ms/step - loss: 0.1425 -
accuracy: 0.9546 - val_loss: 0.3171 - val_accuracy: 0.9052
Epoch 46/60
281/281 [==============================] - 118s 418ms/step - loss: 0.1465 -
accuracy: 0.9531 - val_loss: 0.3139 - val_accuracy: 0.9065
Epoch 47/60
281/281 [==============================] - 116s 413ms/step - loss: 0.1480 -
accuracy: 0.9529 - val_loss: 0.2613 - val_accuracy: 0.9165
Epoch 48/60
281/281 [==============================] - 114s 406ms/step - loss: 0.1512 -
accuracy: 0.9539 - val_loss: 0.2615 - val_accuracy: 0.9324
Epoch 49/60
281/281 [==============================] - 114s 406ms/step - loss: 0.1450 -
accuracy: 0.9558 - val_loss: 0.2617 - val_accuracy: 0.9243
Epoch 50/60
281/281 [==============================] - 117s 415ms/step - loss: 0.1535 -
accuracy: 0.9526 - val_loss: 0.2505 - val_accuracy: 0.9284
Epoch 51/60
281/281 [==============================] - 116s 411ms/step - loss: 0.1430 -
accuracy: 0.9564 - val_loss: 0.2140 - val_accuracy: 0.9344
Epoch 52/60
281/281 [==============================] - 114s 405ms/step - loss: 0.1474 -
accuracy: 0.9563 - val_loss: 0.2797 - val_accuracy: 0.9199
Epoch 53/60
281/281 [==============================] - 115s 410ms/step - loss: 0.1425 -
accuracy: 0.9572 - val_loss: 0.2915 - val_accuracy: 0.9074
Epoch 54/60
281/281 [==============================] - 115s 409ms/step - loss: 0.1470 -
accuracy: 0.9568 - val_loss: 0.1852 - val_accuracy: 0.9453
```

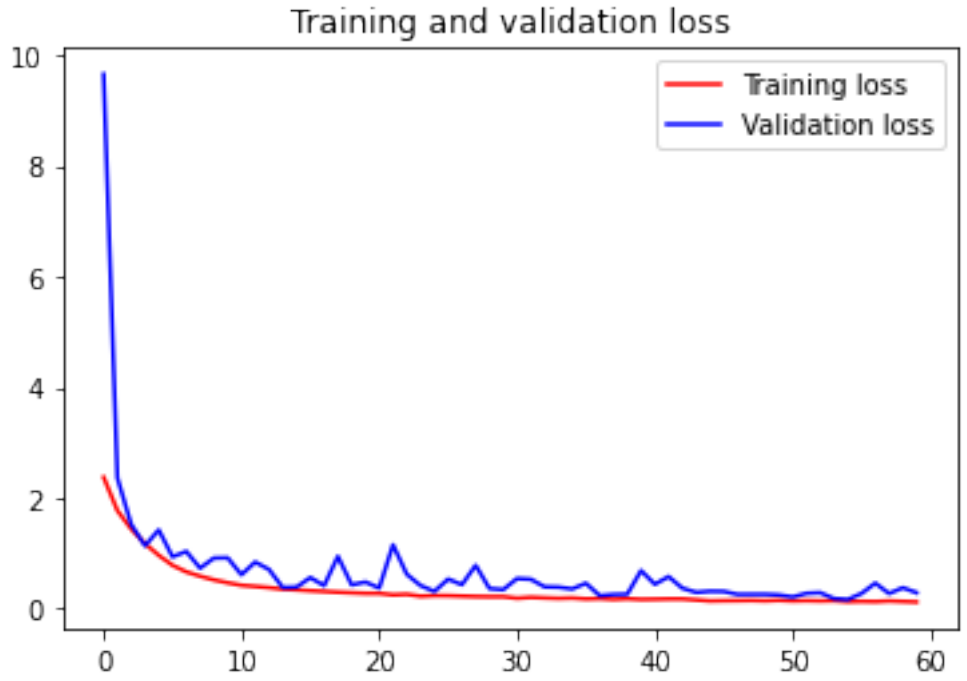
```
Epoch 55/60
281/281 [==============================] - 115s 409ms/step - loss: 0.1298 -
accuracy: 0.9589 - val_loss: 0.1680 - val_accuracy: 0.9543
Epoch 56/60
281/281 [==============================] - 115s 407ms/step - loss: 0.1350 -
accuracy: 0.9583 - val_loss: 0.2776 - val_accuracy: 0.9110
Epoch 57/60
281/281 [==============================] - 115s 409ms/step - loss: 0.1307 -
accuracy: 0.9595 - val_loss: 0.4642 - val_accuracy: 0.9016
Epoch 58/60
281/281 [==============================] - 115s 410ms/step - loss: 0.1422 -
accuracy: 0.9569 - val_loss: 0.2760 - val_accuracy: 0.9177
Epoch 59/60
281/281 [==============================] - 115s 408ms/step - loss: 0.1331 -
accuracy: 0.9596 - val_loss: 0.3813 - val_accuracy: 0.9027
Epoch 60/60
281/281 [==============================] - 115s 407ms/step - loss: 0.1243 -
accuracy: 0.9631 - val_loss: 0.2915 - val_accuracy: 0.9223
```

1.0.6 Plot to show training accuracy vs validation accuracy

[6]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```

1.0.7 Test data prediction

[49]:
```
testparentdirectory = os.path.join(workingdir + '/
 ↪state-farm-distracted-driver-detection/imgs/')
testdataimage = ImageDataGenerator(rescale = 1./255)
testdata = testdataimage.flow_from_directory(testparentdirectory,␣
 ↪classes=['test'], target_size = (100,100))
testoutput = cnnmodel.predict(testdata, verbose = 1)
```

```
Found 79726 images belonging to 1 classes.
2492/2492 [==============================] - 247s 99ms/step
```

1.0.8 Preparing output dataframe

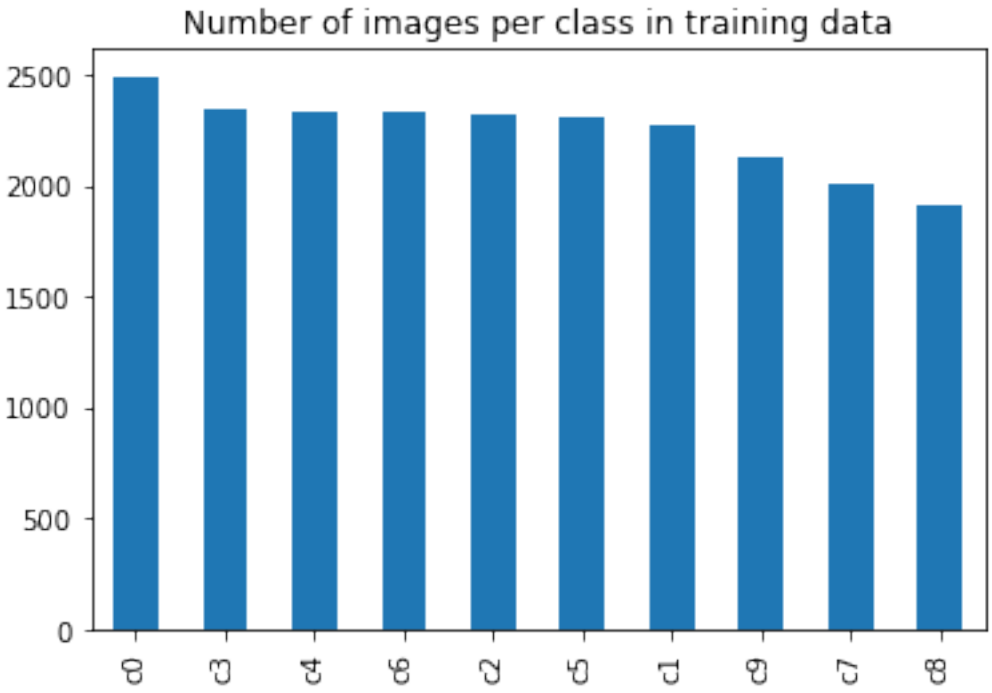
[50]:
```
specimencsv = pd.read_csv(os.path.join(workingdir + '/
 ↪state-farm-distracted-driver-detection/sample_submission.csv'))
result = {'img':list(specimencsv.values[:,0]),}
for value in range(0,10):
    result['c' + str(value)] = list(testoutput[:,value])
```

[53]:
```
testoutput = pd.DataFrame(result)
```

1.0.9 Exploratory data analysis

[57]:
```
imagescsv = pd.read_csv(os.path.join(workingdir + '/
 ↪state-farm-distracted-driver-detection/driver_imgs_list.csv'))
imagescsv.classname.value_counts().plot(kind = 'bar', label = 'index')
plt.title('Number of images per class in training data')
```

[57]:
```
Text(0.5, 1.0, 'Number of images per class in training data')
```



1.0.10 Preparing output file

[60]:
```
testoutput.to_csv('Testoutput.csv', index = False, encoding='utf-8')
```