In [1]:
```
#imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import string
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import plot_confusion_matrix
from sklearn.svm import SVC, NuSVC
from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from scipy.stats import norm
import glob

#imports
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
from gensim.models import Word2Vec
import re
import cv2
import os
import sys

import string
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import plot_confusion_matrix
from sklearn.svm import SVC, NuSVC
from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier
from scipy.stats import norm
from sklearn import tree

import glob

from keras.models import Sequential
from keras_preprocessing.image import ImageDataGenerator
from keras.layers import Dense, Activation, Flatten, Dropout, BatchNormalization
from keras.layers import Conv2D, MaxPooling2D
from keras import regularizers, optimizers

import matplotlib.pyplot as plt
import keras
from keras import layers
from PIL import Image
from sklearn.preprocessing import PolynomialFeatures
import datetime

import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorboard import notebook
from tensorflow.keras.preprocessing.image import Iterator
```

In [2]:
```
train = []
train_gnd  = []
test = []
test_gnd = []
files  = glob.glob('../256_ObjectCategories/*/*.jpg')

j = 0
# for i in range (0,len(files)):
for i in range (0,len(files),5):


  if j%3==0:

    im = keras.preprocessing.image.load_img(files[i],target_size=(268,160))
    im = keras.preprocessing.image.img_to_array(im)
    test_gnd.append(files[i][-12:-9])
    test.append(im)
    j=1

  else:
    im = keras.preprocessing.image.load_img(files[i],target_size=(268,160))
    im = keras.preprocessing.image.img_to_array(im)
    train_gnd.append(files[i][-12:-9])
    train.append(im)
    j=j+1
  #print(i)
print("done")
```

```
done
```

In [3]:
```
print(np.shape(test))
```

```
(2041, 268, 160, 3)
```

In [4]:
```
train_gnd = np.array(train_gnd)
train = np.array(train)
test = np.array(test)
test_gnd = np.array(test_gnd)
train = train.astype('float32') / 255
# train_gnd = train_gnd.astype('float32') / 255
# test_gnd = test_gnd.astype('float32') / 255
test = test.astype('float32')/255

ni,nx,ny,nc=train.shape
d2_train=train.reshape(ni,nx*ny*nc)
ni,nx,ny,nc=test.shape
d2_test=test.reshape(ni,nx*ny*nc)
```

In [5]:
```
def eval_model(model, X_train, Y_train, X_test, Y_test):
    pred = model.predict(X_train)
    print('Training Set Performance: ' + str(sum(pred == Y_train)/len(Y_train)));
    pred = model.predict(X_test)
    print('Testing Set Performance: ' + str(sum(pred == Y_test)/len(Y_test)));
```

In [ ]:
```
#k_nearest
cknn = KNeighborsClassifier(n_neighbors=3, weights='uniform')
cknn.fit(d2_train, train_gnd)
eval_model(cknn, d2_train, train_gnd, d2_test, test_gnd)
```

In [ ]:
```
#random_forest
rf = RandomForestClassifier(n_estimators=256, max_depth=100 ,random_state=0)
rf.fit(d2_train, train_gnd)
eval_model(rf, d2_train, train_gnd, d2_test, test_gnd)
```

In [11]:
```
#k_nearest
cknn = KNeighborsClassifier(n_neighbors=, weights='uniform')
cknn.fit(d2_train, train_gnd)
eval_model(cknn, d2_train, train_gnd, d2_test, test_gnd)
```

```
Training Set Performance: 0.08845871110022054
Testing Set Performance: 0.08280254777070063
```

In [6]:
```
svm = SVC(C=0.1, kernel='poly')
svm.fit(d2_train, train_gnd)
```

Out[6]:
```
SVC(C=0.1, kernel='poly')
```

In [7]:
```
eval_model(svm, d2_train, train_gnd, d2_test, test_gnd)
```

```
Training Set Performance: 0.5726537613330066
Testing Set Performance: 0.10730034296913278
```

In [8]:
```
svm = SVC(C=1, kernel='poly')
svm.fit(d2_train, train_gnd)
```

Out[8]:
```
SVC(C=1, kernel='poly')
```

In [9]:
```
eval_model(svm, d2_train, train_gnd, d2_test, test_gnd)
```

```
Training Set Performance: 0.9259985297721147
Testing Set Performance: 0.11513963743263106
```

In [ ]:
```

```