Dataset preparation

Importing

In [109]:
```
#join google-drive to the code
from google.colab import drive #to connect google drive to the code
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remo
unt=True).
```

In [0]:
```
from keras.layers import Input, Dense
from keras.models import Model
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
from IPython.display import Image
import pandas as pd
import numpy as np
import random
import math
```

Train Db

In [0]:
```
columns=['aeroplane','bicycle','bird','boat','bottle','bus','car','cat','chair',
         'cow','diningtable','dog','horse','motorbike','person','pottedplant',
        'sheep','sofa','train','tvmonitor']

allColumn_lists = []
for item in columns:
    f = open('/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Main/'+item+'_train.txt', 'r')
    item_list = f.read().splitlines()
    f.close()
    item_list = [e for e in item_list if ('-' not in e)]
    column_list = []
    for e in item_list:
      e = e.split(" ", 1)[0]
      column_list.append(e)
    allColumn_lists.append(column_list)
```

In [112]:
```
merged = []
print('Number of Samples:\n-------------------------')
for n in range(len(allColumn_lists)):
    print(columns[n]+' = '+str(len(allColumn_lists[n])))
    merged = merged+allColumn_lists[n]

print('-------------------------')
print('Total number of samples = '+str(len(merged)))
merged_unique = list(set(merged))

print('Total number of Unique samples = '+str(len(merged_unique)))
merged_unique = random.sample(merged_unique, len(merged_unique))
```

```
Number of Samples:
-------------------------
aeroplane = 113
bicycle = 122
bird = 182
boat = 87
bottle = 153
bus = 100
car = 402
cat = 166
chair = 282
cow = 71
diningtable = 130
dog = 210
horse = 144
motorbike = 123
person = 1070
pottedplant = 153
sheep = 49
sofa = 188
train = 128
tvmonitor = 144
-------------------------
Total number of samples = 4017
Total number of Unique samples = 2501
```

In [0]:
```
all_bin_columns = []
match_found = 0

for column in allColumn_lists:
    bin_column =[]
    for merged_file in merged_unique:
        for column_file in column:
            if(merged_file == column_file):  match_found = 1

        if(match_found == 1):
          bin_column.append(1)
          match_found = 0
        else: bin_column.append(0)
    all_bin_columns.append(bin_column)
```

In [0]:
```
fileNames = []
for m in merged_unique:
  m = m+'.jpg'
  fileNames.append(m)
```

In [115]:
```
imageMap_train = pd.DataFrame(
    {'Filenames': fileNames
    })
for n in range(len(columns)):
    imageMap_train[columns[n]] =  all_bin_columns[n]

pd.set_option('display.max_columns', None)
imageMap_train.head(10)
```

Out[115]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	002547.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	006506.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
2	009845.jpg	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	
3	004710.jpg	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
4	002684.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
5	004801.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
6	008970.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
7	007483.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
8	008517.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
9	001263.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	

Validation Db

In [116]:
```
allColumn_lists = []
for item in columns:
    f = open('/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Main/'+item+'_val.txt', 'r')
    item_list = f.read().splitlines()
    f.close()
    item_list = [e for e in item_list if ('-' not in e)]
    column_list = []
    for e in item_list:
      e = e.split(" ", 1)[0]
      column_list.append(e)
    allColumn_lists.append(column_list)

merged = []
print('Number of Samples:\n-------------------------')
for n in range(len(allColumn_lists)):
    print(columns[n]+' = '+str(len(allColumn_lists[n])))
    merged = merged+allColumn_lists[n]

print('--------------------------')
print('Total number of samples = '+str(len(merged)))
merged_unique = list(set(merged))

print('Total number of Unique samples = '+str(len(merged_unique)))
merged_unique = random.sample(merged_unique, len(merged_unique))
```

```
Number of Samples:
-------------------------
aeroplane = 127
bicycle = 133
bird = 151
boat = 101
bottle = 109
bus = 97
car = 359
cat = 178
chair = 290
cow = 75
diningtable = 133
dog = 220
horse = 150
motorbike = 126
person = 1025
pottedplant = 120
sheep = 48
sofa = 184
train = 135
tvmonitor = 135
--------------------------
Total number of samples = 3896
Total number of Unique samples = 2510
```

In [0]:
```
all_bin_columns = []
match_found = 0

for column in allColumn_lists:
    bin_column =[]
    for merged_file in merged_unique:
        for column_file in column:
            if(merged_file == column_file):  match_found = 1

        if(match_found == 1):
          bin_column.append(1)
          match_found = 0
        else: bin_column.append(0)
    all_bin_columns.append(bin_column)
```

In [0]:
```
fileNames = []
for m in merged_unique:
  m = m+'.jpg'
  fileNames.append(m)
```

In [119]:
```
imageMap_valid = pd.DataFrame(
    {'Filenames': fileNames
    })
for n in range(len(columns)):
    imageMap_valid[columns[n]] =  all_bin_columns[n]

pd.set_option('display.max_columns', None)
imageMap_valid.head(10)
```

Out[119]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	006681.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
1	006520.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	008293.jpg	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	
3	003638.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	
4	003793.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
5	000113.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
6	004409.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
7	000424.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	005185.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
9	001160.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [120]:
```
L1 = list(imageMap_valid['Filenames'])
L2 = list(imageMap_train['Filenames'])
if([i for i in L1 if i in L2] == []): print('null')
```

```
null
```

TrainValDb

In [121]:
```
allColumn_lists = []
for item in columns:
    f = open('/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Main/'+item+'_trainval.txt', 'r')
    item_list = f.read().splitlines()
    f.close()
    item_list = [e for e in item_list if ('-' not in e)]
    column_list = []
    for e in item_list:
      e = e.split(" ", 1)[0]
      column_list.append(e)
    allColumn_lists.append(column_list)

merged = []
print('Number of Samples:\n-------------------------')
for n in range(len(allColumn_lists)):
    print(columns[n]+' = '+str(len(allColumn_lists[n])))
    merged = merged+allColumn_lists[n]

print('--------------------------')
print('Total number of samples = '+str(len(merged)))
merged_unique = list(set(merged))

print('Total number of Unique samples = '+str(len(merged_unique)))
merged_unique = random.sample(merged_unique, len(merged_unique))
```

```
Number of Samples:
-------------------------
aeroplane = 240
bicycle = 255
bird = 333
boat = 188
bottle = 262
bus = 197
car = 761
cat = 344
chair = 572
cow = 146
diningtable = 263
dog = 430
horse = 294
motorbike = 249
person = 2095
pottedplant = 273
sheep = 97
sofa = 372
train = 263
tvmonitor = 279
--------------------------
Total number of samples = 7913
Total number of Unique samples = 5011
```

In [0]:
```
all_bin_columns = []
match_found = 0

for column in allColumn_lists:
    bin_column =[]
    for merged_file in merged_unique:
        for column_file in column:
            if(merged_file == column_file):  match_found = 1

        if(match_found == 1):
          bin_column.append(1)
          match_found = 0
        else: bin_column.append(0)
    all_bin_columns.append(bin_column)
```

In [0]:
```
fileNames = []
for m in merged_unique:
  m = m+'.jpg'
  fileNames.append(m)
```

In [124]:
```
imageMap_trainval = pd.DataFrame(
    {'Filenames': fileNames
    })
for n in range(len(columns)):
    imageMap_trainval[columns[n]] =  all_bin_columns[n]

pd.set_option('display.max_columns', None)
imageMap_trainval.head(10)
```

Out[124]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	008141.jpg	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	
1	007460.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	009460.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
3	005755.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	002695.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
5	007650.jpg	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
6	002436.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
7	008607.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
8	000519.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	008091.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	

TestDb

In [125]:
```
allColumn_lists = []
for item in columns:
    f = open('/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Main/'+item+'_test.txt', 'r')
    item_list = f.read().splitlines()
    f.close()
    item_list = [e for e in item_list if ('-' not in e)]
    column_list = []
    for e in item_list:
      e = e.split(" ", 1)[0]
      column_list.append(e)
    allColumn_lists.append(column_list)

merged = []
print('Number of Samples:\n-------------------------')
for n in range(len(allColumn_lists)):
    print(columns[n]+' = '+str(len(allColumn_lists[n])))
    merged = merged+allColumn_lists[n]

print('--------------------------')
print('Total number of samples = '+str(len(merged)))
merged_unique = list(set(merged))

print('Total number of Unique samples = '+str(len(merged_unique)))
merged_unique = random.sample(merged_unique, len(merged_unique))
```

```
Number of Samples:
-------------------------
aeroplane = 205
bicycle = 250
bird = 289
boat = 176
bottle = 240
bus = 183
car = 775
cat = 332
chair = 545
cow = 127
diningtable = 247
dog = 433
horse = 279
motorbike = 233
person = 2097
pottedplant = 254
sheep = 98
sofa = 355
train = 259
tvmonitor = 255
--------------------------
Total number of samples = 7632
Total number of Unique samples = 4952
```

In [0]:
```
all_bin_columns = []
match_found = 0

for column in allColumn_lists:
    bin_column =[]
    for merged_file in merged_unique:
        for column_file in column:
            if(merged_file == column_file):  match_found = 1

        if(match_found == 1):
          bin_column.append(1)
          match_found = 0
        else: bin_column.append(0)
    all_bin_columns.append(bin_column)
```

In [127]:
```
fileNames = []
for m in merged_unique:
  m = m+'.jpg'
  fileNames.append(m)

imageMap_test = pd.DataFrame(
    {'Filenames': fileNames
    })
for n in range(len(columns)):
    imageMap_test[columns[n]] =  all_bin_columns[n]

pd.set_option('display.max_columns', None)
imageMap_test.head(10)
```

Out[127]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	007157.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
1	002736.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
2	003624.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
3	002106.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
4	002927.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
5	008964.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	
6	004589.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
7	001698.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	008567.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
9	005119.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

Modified TrainValDb

In [128]:
```
#for class[0] or column[0]
pos_column_db = imageMap_trainval[imageMap_trainval[columns[0]] == 1]
pos_column_db = pos_column_db.reset_index(drop=True)
neg_column_db = imageMap_trainval[imageMap_trainval[columns[0]] == 0]
neg_column_db_shuffled = neg_column_db.sample(frac=1).reset_index(drop=True)
neg_column_db_cut = neg_column_db[:len(pos_column_db)*2] #*2 #+50
print('Number of positive samples: ',len(pos_column_db))
print('Number of selected negative samples: ',len(neg_column_db_cut))
modifiedDb = pos_column_db.append(neg_column_db_cut, ignore_index=True)
modifiedDb = modifiedDb.sample(frac=1).reset_index(drop=True)
print('Number of merged samples: ',len(modifiedDb))
modifiedDb[:10]
```

```
Number of positive samples:  240
Number of selected negative samples:  480
Number of merged samples:  720
```

Out[128]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	006281.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	008604.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	
2	001052.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
3	003667.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	005110.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
5	000007.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
6	009408.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	002709.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	003580.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
9	004368.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	

Modified TrainValDb Alternative

In [129]:
```
#for class[0] or column[0]
pos_column_db = imageMap_trainval[imageMap_trainval[columns[0]] == 1]
pos_column_db = pos_column_db.reset_index(drop=True)
neg_column_db = imageMap_trainval[imageMap_trainval[columns[0]] == 0]
neg_column_db_shuffled = neg_column_db.sample(frac=1).reset_index(drop=True)
#neg_column_db_cut = neg_column_db[:len(pos_column_db)*2] #*2 #+50
neg_column_db_cut = neg_column_db[:2] #*2 #+50

fraction = int((len(pos_column_db)*5)/19)

for n in range(len(columns)):
    if(columns[n] == columns[0]): print('Avoided :',columns[0])
    else:
      one_neg_sample_db = neg_column_db[neg_column_db[columns[n]] == 1]
      one_neg_sample_db = one_neg_sample_db.sample(frac=1)[:fraction]
      neg_column_db_cut = neg_column_db_cut.append(one_neg_sample_db)
      neg_column_db_cut = neg_column_db_cut.sample(frac=1)
      neg_column_db_cut = neg_column_db_cut.reset_index(drop=True)

neg_column_db_cut[:10]
```

```
Avoided : aeroplane
```

Out[129]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	005517.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	
1	002082.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
2	007466.jpg	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	
3	008722.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
4	004694.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
5	000311.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
6	008564.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
7	004911.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	006760.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
9	002798.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	

In [130]:
```
print('Number of positive samples: ',len(pos_column_db))
print('Number of selected negative samples: ',len(neg_column_db_cut))
modifiedDb2 = pos_column_db.append(neg_column_db_cut, ignore_index=True)
modifiedDb2 = modifiedDb2.sample(frac=1).reset_index(drop=True)
print('Number of merged samples: ',len(modifiedDb2))
modifiedDb2[:10]
```

```
Number of positive samples:  240
Number of selected negative samples:  1199
Number of merged samples:  1439
```

Out[130]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	008204.jpg	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	
1	007845.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	007329.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
3	004110.jpg	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	
4	005373.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
5	004878.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
6	003365.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	002578.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
8	005160.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	
9	007723.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

Model 01

Training Column 0

Training with modified TrainVal Db

In [0]:
```
training_columns = [columns[0]]

datagen=ImageDataGenerator(rescale=1./255.)
test_datagen=ImageDataGenerator(rescale=1./255.)

print('For Training:')
train_generator=datagen.flow_from_dataframe(
    dataframe=modifiedDb[:-50],#modifiedDb[:680], #490
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Validation:')
valid_generator=test_datagen.flow_from_dataframe(
    dataframe=modifiedDb[-50:],#modifiedDb[680:],  #490
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Testing:')
test_generator=test_datagen.flow_from_dataframe(
    dataframe=imageMap_test,
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images",
    x_col="Filenames",
    batch_size=1,
    seed=42,
    shuffle=False,
    class_mode=None,
    target_size=(100,100))
```

```
For Training:
Found 670 images.
For Validation:
Found 50 images.
For Testing:
Found 4952 images.
```

In [0]:
```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=(100,100,3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))  #Depends on number of classes
model.compile(optimizers.rmsprop(lr=0.0001, decay=1e-6),loss="binary_crossentropy",metrics=["accuracy"])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/op_def_library.py:263: colo
cate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3445: calling dropo
ut (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

In [0]:
```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/10
20/20 [==============================] - 164s 8s/step - loss: 0.6363 - acc: 0.6469 - val_loss: 0.6393 - val_acc: 0.500
0
Epoch 2/10
20/20 [==============================] - 50s 2s/step - loss: 0.4613 - acc: 0.7994 - val_loss: 0.4091 - val_acc: 0.8889
Epoch 3/10
20/20 [==============================] - 44s 2s/step - loss: 0.4589 - acc: 0.7964 - val_loss: 0.4691 - val_acc: 0.8750
Epoch 4/10
20/20 [==============================] - 44s 2s/step - loss: 0.4177 - acc: 0.8230 - val_loss: 0.4483 - val_acc: 0.8333
Epoch 5/10
20/20 [==============================] - 44s 2s/step - loss: 0.4279 - acc: 0.8148 - val_loss: 0.5717 - val_acc: 0.6875
Epoch 6/10
20/20 [==============================] - 44s 2s/step - loss: 0.3642 - acc: 0.8350 - val_loss: 0.5097 - val_acc: 0.8889
Epoch 7/10
20/20 [==============================] - 44s 2s/step - loss: 0.4215 - acc: 0.8242 - val_loss: 0.4869 - val_acc: 0.7500
Epoch 8/10
20/20 [==============================] - 44s 2s/step - loss: 0.4097 - acc: 0.8323 - val_loss: 0.4863 - val_acc: 0.6667
Epoch 9/10
20/20 [==============================] - 44s 2s/step - loss: 0.3430 - acc: 0.8606 - val_loss: 0.4679 - val_acc: 0.7188
Epoch 10/10
20/20 [==============================] - 44s 2s/step - loss: 0.3610 - acc: 0.8574 - val_loss: 0.6675 - val_acc: 0.8333
```

Out[0]:
```
<keras.callbacks.History at 0x7fe177ec0da0>
```

In [0]:
```
test_generator.reset()
pred=model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)
```

```
4952/4952 [==============================] - 1192s 241ms/step
```

In [0]:
```
pred_bool = (pred >0.5)

predictions = pred_bool.astype(int)
testing_columns = [columns[0]]

#columns should be the same order of y_col
results = pd.DataFrame(predictions, columns = testing_columns)
results["Filenames"] = test_generator.filenames
#results["Filenames"] = test_generator.filenames[30]
ordered_cols = ["Filenames"] + testing_columns
results = results[ordered_cols]#To get the same column order
print('Predicted Results (for ones only)---------------')
results[results[columns[0]] == 1][:5]
#results
```

```
Predicted Results (for ones only)---------------
```

Out[0]:

	Filenames	aeroplane
3	000924.jpg	1
4	003910.jpg	1
16	005333.jpg	1
24	009553.jpg	1
35	002968.jpg	1

In [0]:
```
predicted_ones_df = results[results[columns[0]] == 1]
actual_ones_df = imageMap_test[imageMap_test[columns[0]] == 1]

predicted = list(predicted_ones_df['Filenames'])
actual = list(actual_ones_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('Samples of rightly predicted as one\n----------------------')
print(total_match[:5])

print('actual ones: ',len(actual))
print('predicted ones: ',len(predicted))
print('Matched ones(True Positive): ', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of ones: ',prediction_accuracy,'%')

predicted_o_df = results[results[columns[0]] == 0]
actual_o_df = imageMap_test[imageMap_test[columns[0]] == 0]

predicted = list(predicted_o_df['Filenames'])
actual = list(actual_o_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('\n\nSamples of rightly predicted as zero\n----------------------')
print(total_match[:5])

print('actual zeros: ',len(actual))
print('predicted zeros: ',len(predicted))
print('Matched zeros (True Negative): ', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of zeros: ',prediction_accuracy,'%')
```

```
predicted = np.array(results[columns[0]])
actual = np.array(imageMap_test[columns[0]])

total_match = np.sum(actual == predicted)
print('\n\nRightly predicted overall\n----------------------')
prediction_accuracy = (total_match / len(actual))*100
print('Number of samples', len(actual))
print('Overall Matches', total_match)
print('Overall prediction accuracy: ',prediction_accuracy,'%')
```

```
Samples of rightly predicted as one
----------------------
['009553.jpg', '007164.jpg', '009876.jpg', '009262.jpg', '007286.jpg']
actual ones:  205
predicted ones:  319
Matched ones(True Positive):  119
prediction accuracy of ones:  58.048780487804876 %


Samples of rightly predicted as zero
----------------------
['008340.jpg', '003734.jpg', '009572.jpg', '001957.jpg', '003399.jpg']
actual zeros:  4747
predicted zeros:  4633
Matched zeros (True Negative):  4547
prediction accuracy of zeros:  95.78681272382558 %


Rightly predicted overall
----------------------
Number of samples 4952
Overall Matches 4666
Overall prediction accuracy:  94.22455573505655 %
```

Training with modified TrainVal Db Alternative

In [0]:
```
training_columns = [columns[0]]

datagen=ImageDataGenerator(rescale=1./255.)
test_datagen=ImageDataGenerator(rescale=1./255.)

print('For Training:')
train_generator=datagen.flow_from_dataframe(
    dataframe=modifiedDb2[:-100],#modifiedDb[:680], #490
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Validation:')
valid_generator=test_datagen.flow_from_dataframe(
    dataframe=modifiedDb2[-100:],#modifiedDb[680:],  #490
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Testing:')
test_generator=test_datagen.flow_from_dataframe(
    dataframe=imageMap_test,
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images",
    x_col="Filenames",
    batch_size=1,
    seed=42,
    shuffle=False,
    class_mode=None,
    target_size=(100,100))
```

```
For Training:
Found 1271 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

In [0]:
```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=(100,100,3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))


model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))  #Depends on number of classes
model.compile(optimizers.rmsprop(lr=0.0001, decay=1e-6),loss="binary_crossentropy",metrics=["accuracy"])
```

In [0]:
```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
```

```
Epoch 1/20
39/39 [==============================] - 88s 2s/step - loss: 0.5222 - acc: 0.8213 - val_loss: 0.5023 - val_acc: 0.7917
Epoch 2/20
39/39 [==============================] - 83s 2s/step - loss: 0.3756 - acc: 0.8436 - val_loss: 0.4262 - val_acc: 0.8676
Epoch 3/20
39/39 [==============================] - 82s 2s/step - loss: 0.3292 - acc: 0.8580 - val_loss: 0.4650 - val_acc: 0.7794
Epoch 4/20
39/39 [==============================] - 82s 2s/step - loss: 0.3378 - acc: 0.8625 - val_loss: 0.3194 - val_acc: 0.9412
Epoch 5/20
39/39 [==============================] - 83s 2s/step - loss: 0.3046 - acc: 0.8707 - val_loss: 0.3342 - val_acc: 0.8958
Epoch 6/20
39/39 [==============================] - 82s 2s/step - loss: 0.3043 - acc: 0.8824 - val_loss: 0.4463 - val_acc: 0.7794
Epoch 7/20
39/39 [==============================] - 82s 2s/step - loss: 0.2982 - acc: 0.8762 - val_loss: 0.3271 - val_acc: 0.9265
Epoch 8/20
39/39 [==============================] - 82s 2s/step - loss: 0.2782 - acc: 0.8941 - val_loss: 0.3796 - val_acc: 0.8529
Epoch 9/20
39/39 [==============================] - 83s 2s/step - loss: 0.2757 - acc: 0.9019 - val_loss: 0.3001 - val_acc: 0.8958
Epoch 10/20
39/39 [==============================] - 83s 2s/step - loss: 0.2538 - acc: 0.9056 - val_loss: 0.3678 - val_acc: 0.8235
Epoch 11/20
39/39 [==============================] - 82s 2s/step - loss: 0.2582 - acc: 0.9051 - val_loss: 0.2743 - val_acc: 0.9118
Epoch 12/20
39/39 [==============================] - 83s 2s/step - loss: 0.2757 - acc: 0.8872 - val_loss: 0.3170 - val_acc: 0.8824
Epoch 13/20
39/39 [==============================] - 83s 2s/step - loss: 0.2230 - acc: 0.9171 - val_loss: 0.2828 - val_acc: 0.8750
Epoch 14/20
39/39 [==============================] - 83s 2s/step - loss: 0.2666 - acc: 0.9000 - val_loss: 0.3572 - val_acc: 0.8824
Epoch 15/20
39/39 [==============================] - 83s 2s/step - loss: 0.2410 - acc: 0.9147 - val_loss: 0.2314 - val_acc: 0.9118
Epoch 16/20
39/39 [==============================] - 82s 2s/step - loss: 0.2286 - acc: 0.9115 - val_loss: 0.2814 - val_acc: 0.8676
Epoch 17/20
39/39 [==============================] - 83s 2s/step - loss: 0.2382 - acc: 0.9157 - val_loss: 0.2591 - val_acc: 0.9062
Epoch 18/20
39/39 [==============================] - 83s 2s/step - loss: 0.2411 - acc: 0.9007 - val_loss: 0.2105 - val_acc: 0.8971
Epoch 19/20
39/39 [==============================] - 82s 2s/step - loss: 0.2352 - acc: 0.9039 - val_loss: 0.3015 - val_acc: 0.9118
Epoch 20/20
39/39 [==============================] - 82s 2s/step - loss: 0.2174 - acc: 0.9186 - val_loss: 0.3412 - val_acc: 0.8382
```

Out[0]:
```
<keras.callbacks.History at 0x7f479b1a31d0>
```

In [0]:
```
test_generator.reset()
pred2 = model.predict_generator(test_generator,
                                steps=STEP_SIZE_TEST,
                                #steps = 100,
                                verbose=1)
```

```
4952/4952 [==============================] - 179s 36ms/step
```

In [0]:
```
pred_bool = (pred2 >0.5)
predictions = pred_bool.astype(int)
testing_columns = [columns[0]]

#columns should be the same order of y_col
results2 = pd.DataFrame(predictions, columns = testing_columns)
results2["Filenames"] = test_generator.filenames
#results["Filenames"] = test_generator.filenames[30]
ordered_cols = ["Filenames"] + testing_columns
results2 = results2[ordered_cols]#To get the same column order
print('Predicted Results (for ones only)---------------')
results2[results2[columns[0]] == 1][:5]
#results
```

```
Predicted Results (for ones only)---------------
```

Out[0]:

	Filenames	aeroplane
3	000994.jpg	1
5	006716.jpg	1
9	002703.jpg	1
26	007993.jpg	1
35	003850.jpg	1

In [0]:
```
predicted_ones_df = results2[results2[columns[0]] == 1]
actual_ones_df = imageMap_test[imageMap_test[columns[0]] == 1]

predicted = list(predicted_ones_df['Filenames'])
actual = list(actual_ones_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('Samples of rightly predicted as one\n----------------------')
print(total_match[:5])

print('actual ones: ',len(actual))
print('predicted ones: ',len(predicted))
print('Matched ones(True Positive): ', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of ones: ',prediction_accuracy,'%')

predicted_o_df = results2[results2[columns[0]] == 0]
actual_o_df = imageMap_test[imageMap_test[columns[0]] == 0]

predicted = list(predicted_o_df['Filenames'])
actual = list(actual_o_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('\n\nSamples of rightly predicted as zero\n----------------------')
print(total_match[:5])

print('actual zeros: ',len(actual))
print('predicted zeros: ',len(predicted))
print('Matched zeros (True Negative): ', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of zeros: ',prediction_accuracy,'%')


predicted = np.array(results2[columns[0]])
actual = np.array(imageMap_test[columns[0]])

total_match = np.sum(actual == predicted)
print('\n\nRightly predicted overall\n----------------------')
prediction_accuracy = (total_match / len(actual))*100
print('Number of samples', len(actual))
print('Overall Matches', total_match)
print('Overall prediction accuracy: ',prediction_accuracy,'%')
```

```
Samples of rightly predicted as one
----------------------
['002703.jpg', '007993.jpg', '006406.jpg', '009838.jpg', '000316.jpg']
actual ones:  205
predicted ones:  808
Matched ones(True Positive):  154
prediction accuracy of ones:  75.1219512195122 %


Samples of rightly predicted as zero
----------------------
['001966.jpg', '003029.jpg', '001600.jpg', '000097.jpg', '003378.jpg']
actual zeros:  4747
predicted zeros:  4144
Matched zeros (True Negative):  4093
prediction accuracy of zeros:  86.22287760690963 %


Rightly predicted overall
----------------------
Number of samples 4952
Overall Matches 4247
Overall prediction accuracy:  85.76332794830371 %
```

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c0.csv')
```

Training with TrainDb

In [0]:
```
training_columns = [columns[0]]

datagen=ImageDataGenerator(rescale=1./255.)
test_datagen=ImageDataGenerator(rescale=1./255.)

test_generator.reset()
train_generator.reset()
valid_generator.reset()

print('For Training')
train_generator=datagen.flow_from_dataframe(
    dataframe=imageMap_train,
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Validation')
valid_generator=test_datagen.flow_from_dataframe(
    dataframe=imageMap_valid,
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Testing')
test_generator=test_datagen.flow_from_dataframe(
    dataframe=imageMap_test,
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images",
    x_col="Filenames",
    batch_size=1,
    seed=42,
    shuffle=False,
    class_mode=None,
    target_size=(100,100))
```

```
For Training
Found 2501 images.
For Validation
Found 2510 images.
For Testing
Found 4952 images.
```

In [0]:
```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=(100,100,3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))  #Depends on number of classes
model.compile(optimizers.rmsprop(lr=0.0001, decay=1e-6),loss="binary_crossentropy",metrics=["accuracy"])
```

In [0]:
```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
```

```
Epoch 1/10
78/78 [==============================] - 787s 10s/step - loss: 0.1880 - acc: 0.9479 - val_loss: 0.2277 - val_acc: 0.94
95
Epoch 2/10
78/78 [==============================] - 210s 3s/step - loss: 0.1412 - acc: 0.9551 - val_loss: 0.1971 - val_acc: 0.948
7
Epoch 3/10
78/78 [==============================] - 208s 3s/step - loss: 0.1279 - acc: 0.9559 - val_loss: 0.2611 - val_acc: 0.940
7
Epoch 4/10
78/78 [==============================] - 208s 3s/step - loss: 0.1372 - acc: 0.9554 - val_loss: 0.1767 - val_acc: 0.954
4
Epoch 5/10
78/78 [==============================] - 209s 3s/step - loss: 0.1176 - acc: 0.9631 - val_loss: 0.2313 - val_acc: 0.924
9
Epoch 6/10
78/78 [==============================] - 208s 3s/step - loss: 0.1160 - acc: 0.9643 - val_loss: 0.1324 - val_acc: 0.954
8
Epoch 7/10
78/78 [==============================] - 209s 3s/step - loss: 0.1193 - acc: 0.9611 - val_loss: 0.1449 - val_acc: 0.954
4
Epoch 8/10
78/78 [==============================] - 208s 3s/step - loss: 0.1086 - acc: 0.9635 - val_loss: 0.1313 - val_acc: 0.954
8
Epoch 9/10
78/78 [==============================] - 208s 3s/step - loss: 0.1131 - acc: 0.9643 - val_loss: 0.1424 - val_acc: 0.955
2
Epoch 10/10
78/78 [==============================] - 208s 3s/step - loss: 0.1164 - acc: 0.9667 - val_loss: 0.1627 - val_acc: 0.945
9
```

Out[0]:
```
<keras.callbacks.History at 0x7fe175f0f860>
```

In [0]:
```
test_generator.reset()
pred3 = model.predict_generator(test_generator,
                                steps=STEP_SIZE_TEST,
                                #steps = 100,
                                verbose=1)
```

```
4952/4952 [==============================] - 174s 35ms/step
```

In [0]:
```
pred_bool = (pred3 >0.5)

predictions = pred_bool.astype(int)
testing_columns = [columns[0]]

#columns should be the same order of y_col
results3 = pd.DataFrame(predictions, columns = testing_columns)
results3["Filenames"] = test_generator.filenames
#results["Filenames"] = test_generator.filenames[30]
ordered_cols = ["Filenames"] + testing_columns
results3 = results3[ordered_cols]#To get the same column order
print('Predicted Results (for ones only)---------------')
results3[results3[columns[0]] == 1][:5]
#results
```

```
Predicted Results (for ones only)---------------
```

Out[0]:

	Filenames	aeroplane
3	000924.jpg	1
4	003910.jpg	1
24	009553.jpg	1
35	002968.jpg	1
65	005759.jpg	1

In [0]:
```
predicted_ones_df = results3[results3[columns[0]] == 1]
actual_ones_df = imageMap_test[imageMap_test[columns[0]] == 1]

predicted = list(predicted_ones_df['Filenames'])
actual = list(actual_ones_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('Samples of rightly predicted as one\n----------------------')
print(total_match[:5])

print('actual ones',len(actual))
print('predicted ones',len(predicted))
print('Matched ones', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of ones: ',prediction_accuracy,'%')

predicted_o_df = results3[results3[columns[0]] == 0]
actual_o_df = imageMap_test[imageMap_test[columns[0]] == 0]

predicted = list(predicted_o_df['Filenames'])
actual = list(actual_o_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('\n\nSamples of rightly predicted as zero\n----------------------')
print(total_match[:5])

print('actual zeros',len(actual))
print('predicted zeros',len(predicted))
print('Matched zeros', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of zeros: ',prediction_accuracy,'%')
```

```
predicted = np.array(results3[columns[0]])
actual = np.array(imageMap_test[columns[0]])

total_match = np.sum(actual == predicted)
print('\n\nRightly predicted overall\n----------------------')
prediction_accuracy = (total_match / len(actual))*100
print('Number of samples', len(actual))
print('Overall Matches', total_match)
print('Overall prediction accuracy: ',prediction_accuracy,'%')
```

```
Samples of rightly predicted as one
----------------------
['009553.jpg', '009876.jpg', '009262.jpg', '007286.jpg', '006752.jpg']
actual ones 205
predicted ones 222
Matched ones 101
prediction accuracy of ones:  49.26829268292683 %


Samples of rightly predicted as zero
----------------------
['008340.jpg', '003734.jpg', '009572.jpg', '001957.jpg', '003399.jpg']
actual zeros 4747
predicted zeros 4730
Matched zeros 4626
prediction accuracy of zeros:  97.45102169791447 %


Rightly predicted overall
----------------------
Number of samples 4952
Overall Matches 4727
Overall prediction accuracy:  95.45638126009693 %
```

Training with TrainVal Db

In [0]:
```
training_columns = [columns[0]]

datagen=ImageDataGenerator(rescale=1./255.)
test_datagen=ImageDataGenerator(rescale=1./255.)

test_generator.reset()
train_generator.reset()
valid_generator.reset()

print('For Training')
train_generator=datagen.flow_from_dataframe(
    dataframe=imageMap_trainval[:-200],
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Validation')
valid_generator=test_datagen.flow_from_dataframe(
    dataframe=imageMap_trainval[-200:],
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Testing')
test_generator=test_datagen.flow_from_dataframe(
    dataframe=imageMap_test,
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images",
    x_col="Filenames",
    batch_size=1,
    seed=42,
    shuffle=False,
    class_mode=None,
    target_size=(100,100))
```

```
For Training
Found 4811 images.
For Validation
Found 200 images.
For Testing
Found 4952 images.
```

In [0]:
```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=(100,100,3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))  #Depends on number of classes
model.compile(optimizers.rmsprop(lr=0.0001, decay=1e-6),loss="binary_crossentropy",metrics=["accuracy"])
```

In [0]:
```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
```

```
Epoch 1/10
150/150 [==============================] - 321s 2s/step - loss: 0.1764 - acc: 0.9477 - val_loss: 0.2630 - val_acc: 0.9
740
Epoch 2/10
150/150 [==============================] - 315s 2s/step - loss: 0.1418 - acc: 0.9535 - val_loss: 0.1229 - val_acc: 0.9
583
Epoch 3/10
150/150 [==============================] - 315s 2s/step - loss: 0.1387 - acc: 0.9537 - val_loss: 0.1006 - val_acc: 0.9
762
Epoch 4/10
150/150 [==============================] - 314s 2s/step - loss: 0.1292 - acc: 0.9571 - val_loss: 0.0979 - val_acc: 0.9
702
Epoch 5/10
150/150 [==============================] - 309s 2s/step - loss: 0.1284 - acc: 0.9581 - val_loss: 0.1264 - val_acc: 0.9
524
Epoch 6/10
150/150 [==============================] - 310s 2s/step - loss: 0.1292 - acc: 0.9598 - val_loss: 0.1024 - val_acc: 0.9
583
Epoch 7/10
150/150 [==============================] - 311s 2s/step - loss: 0.1210 - acc: 0.9610 - val_loss: 0.0930 - val_acc: 0.9
702
Epoch 8/10
150/150 [==============================] - 315s 2s/step - loss: 0.1234 - acc: 0.9575 - val_loss: 0.1062 - val_acc: 0.9
635
Epoch 9/10
150/150 [==============================] - 314s 2s/step - loss: 0.1212 - acc: 0.9610 - val_loss: 0.1256 - val_acc: 0.9
643
Epoch 10/10
150/150 [==============================] - 314s 2s/step - loss: 0.1149 - acc: 0.9623 - val_loss: 0.0869 - val_acc: 0.9
643
```

Out[0]:
```
<keras.callbacks.History at 0x7fe177daa630>
```

In [0]:
```
test_generator.reset()
pred4 = model.predict_generator(test_generator,
                                steps=STEP_SIZE_TEST,
                                #steps = 100,
                                verbose=1)
```

```
4952/4952 [==============================] - 178s 36ms/step
```

In [0]:
```
pred_bool = (pred4 >0.5)

predictions = pred_bool.astype(int)
testing_columns = [columns[0]]

#columns should be the same order of y_col
results4 = pd.DataFrame(predictions, columns = testing_columns)
results4["Filenames"] = test_generator.filenames
#results["Filenames"] = test_generator.filenames[30]
ordered_cols = ["Filenames"] + testing_columns
results4 = results4[ordered_cols]#To get the same column order
print('Predicted Results (for ones only)---------------')
results4[results4[columns[0]] == 1][:5]
#results
```

```
Predicted Results (for ones only)---------------
```

Out[0]:

	Filenames	aeroplane
24	009553.jpg	1
65	005759.jpg	1
95	007789.jpg	1
234	009876.jpg	1
243	009262.jpg	1

In [0]:
```
predicted_ones_df = results4[results4[columns[0]] == 1]
actual_ones_df = imageMap_test[imageMap_test[columns[0]] == 1]

predicted = list(predicted_ones_df['Filenames'])
actual = list(actual_ones_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('Samples of rightly predicted as one\n----------------------')
print(total_match[:5])

print('actual ones',len(actual))
print('predicted ones',len(predicted))
print('Matched ones', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of ones: ',prediction_accuracy,'%')

predicted_o_df = results4[results4[columns[0]] == 0]
actual_o_df = imageMap_test[imageMap_test[columns[0]] == 0]

predicted = list(predicted_o_df['Filenames'])
actual = list(actual_o_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('\n\nSamples of rightly predicted as zero\n----------------------')
print(total_match[:5])

print('actual zeros',len(actual))
print('predicted zeros',len(predicted))
print('Matched zeros', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of zeros: ',prediction_accuracy,'%')
```

```
predicted = np.array(results4[columns[0]])
actual = np.array(imageMap_test[columns[0]])

total_match = np.sum(actual == predicted)
print('\n\nRightly predicted overall\n----------------------')
prediction_accuracy = (total_match / len(actual))*100
print('Number of samples', len(actual))
print('Overall Matches', total_match)
print('Overall prediction accuracy: ',prediction_accuracy,'%')
```

```
Samples of rightly predicted as one
----------------------
['009553.jpg', '009876.jpg', '009262.jpg', '007286.jpg', '006752.jpg']
actual ones 205
predicted ones 129
Matched ones 85
prediction accuracy of ones:  41.46341463414634 %


Samples of rightly predicted as zero
----------------------
['008340.jpg', '003734.jpg', '009572.jpg', '000924.jpg', '003910.jpg']
actual zeros 4747
predicted zeros 4823
Matched zeros 4703
prediction accuracy of zeros:  99.07309879924162 %


Rightly predicted overall
----------------------
Number of samples 4952
Overall Matches 4788
Overall prediction accuracy:  96.68820678513733 %
```

Training Column 1

In [0]:
```
#for class[1] or column[1]
pos_column_db = imageMap_trainval[imageMap_trainval[columns[1]] == 1]
pos_column_db = pos_column_db.reset_index(drop=True)
neg_column_db = imageMap_trainval[imageMap_trainval[columns[1]] == 0]
neg_column_db_shuffled = neg_column_db.sample(frac=1).reset_index(drop=True)
#neg_column_db_cut = neg_column_db[:len(pos_column_db)*2] #*2 #+50
neg_column_db_cut = neg_column_db[:2] #*2 #+50

fraction = int((len(pos_column_db)*5)/19)

for n in range(len(columns)):
    if(columns[n] == columns[1]): print('Avoided :',columns[1])
    else:
      one_neg_sample_db = neg_column_db[neg_column_db[columns[n]] == 1]
      one_neg_sample_db = one_neg_sample_db.sample(frac=1)[:fraction]
      neg_column_db_cut = neg_column_db_cut.append(one_neg_sample_db)
      neg_column_db_cut = neg_column_db_cut.sample(frac=1)
      neg_column_db_cut = neg_column_db_cut.reset_index(drop=True)

neg_column_db_cut[:10]
```

```
Avoided : bicycle
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	005292.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	007045.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	
2	006251.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
3	005705.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	008883.jpg	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
5	008036.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
6	008130.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	
7	009700.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
8	003117.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
9	005337.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [0]:
```
print('Number of positive samples: ',len(pos_column_db))
print('Number of selected negative samples: ',len(neg_column_db_cut))
modifiedDb2 = pos_column_db.append(neg_column_db_cut, ignore_index=True)
modifiedDb2 = modifiedDb2.sample(frac=1).reset_index(drop=True)
print('Number of merged samples: ',len(modifiedDb2))
modifiedDb2[:10]
```

```
Number of positive samples:  255
Number of selected negative samples:  1275
Number of merged samples:  1530
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	002209.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
1	002723.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
2	002375.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
3	009613.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	
4	002187.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
5	008810.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
6	009456.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
7	005408.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
8	006551.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
9	006626.jpg	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	

In [0]:
```
training_columns = [columns[1]]

datagen=ImageDataGenerator(rescale=1./255.)
test_datagen=ImageDataGenerator(rescale=1./255.)

test_generator.reset()
train_generator.reset()
valid_generator.reset()

print('For Training:')
train_generator=datagen.flow_from_dataframe(
    dataframe=modifiedDb2[:-100],#modifiedDb[:680], #490
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Validation:')
valid_generator=test_datagen.flow_from_dataframe(
    dataframe=modifiedDb2[-100:],#modifiedDb[680:],  #490
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
    x_col="Filenames",
    y_col=training_columns,
    batch_size=32,
    seed=42,
    shuffle=True,
    class_mode="other",
    target_size=(100,100))

print('For Testing:')
test_generator=test_datagen.flow_from_dataframe(
    dataframe=imageMap_test,
    directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images",
    x_col="Filenames",
    batch_size=1,
    seed=42,
    shuffle=False,
    class_mode=None,
    target_size=(100,100))
```

```
For Training:
Found 1375 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

In [0]:
```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                 input_shape=(100,100,3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))  #Depends on number of classes
model.compile(optimizers.rmsprop(lr=0.0001, decay=1e-6),loss="binary_crossentropy",metrics=["accuracy"])
```

In [0]:
```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
```

```
Epoch 1/10
42/42 [==============================] - 94s 2s/step - loss: 0.5196 - acc: 0.8199 - val_loss: 0.4898 - val_acc: 0.8750
Epoch 2/10
42/42 [==============================] - 90s 2s/step - loss: 0.4830 - acc: 0.8228 - val_loss: 0.4844 - val_acc: 0.8971
Epoch 3/10
42/42 [==============================] - 90s 2s/step - loss: 0.4596 - acc: 0.8266 - val_loss: 0.5528 - val_acc: 0.8235
Epoch 4/10
42/42 [==============================] - 90s 2s/step - loss: 0.4527 - acc: 0.8221 - val_loss: 0.4815 - val_acc: 0.8824
Epoch 5/10
42/42 [==============================] - 91s 2s/step - loss: 0.4299 - acc: 0.8236 - val_loss: 0.5505 - val_acc: 0.8021
Epoch 6/10
42/42 [==============================] - 90s 2s/step - loss: 0.4051 - acc: 0.8295 - val_loss: 0.5630 - val_acc: 0.8971
Epoch 7/10
42/42 [==============================] - 91s 2s/step - loss: 0.4119 - acc: 0.8243 - val_loss: 0.5017 - val_acc: 0.7353
Epoch 8/10
42/42 [==============================] - 90s 2s/step - loss: 0.3864 - acc: 0.8318 - val_loss: 0.5416 - val_acc: 0.7647
Epoch 9/10
42/42 [==============================] - 91s 2s/step - loss: 0.3691 - acc: 0.8355 - val_loss: 0.5726 - val_acc: 0.6771
Epoch 10/10
42/42 [==============================] - 91s 2s/step - loss: 0.3751 - acc: 0.8384 - val_loss: 0.5845 - val_acc: 0.6471
```

Out[0]:
```
<keras.callbacks.History at 0x7f4799b94e80>
```

In [0]:
```
test_generator.reset()
pred_b = model.predict_generator(test_generator,
                                 steps=STEP_SIZE_TEST,
                                 #steps = 100,
                                 verbose=1)
```

```
4952/4952 [==============================] - 181s 37ms/step
```

In [0]:
```
pred_bool = (pred_b >0.50)

predictions = pred_bool.astype(int)
testing_columns = [columns[1]]

#columns should be the same order of y_col
results_b = pd.DataFrame(predictions, columns = testing_columns)
results_b["Filenames"] = test_generator.filenames
#results["Filenames"] = test_generator.filenames[30]
ordered_cols = ["Filenames"] + testing_columns
results_b = results_b[ordered_cols]#To get the same column order
print('Predicted Results (for ones only)---------------')
results_b[results_b[columns[1]] == 1][:5]
#results
```

```
Predicted Results (for ones only)---------------
```

Out[0]:

	Filenames	bicycle
0	001966.jpg	1
5	006716.jpg	1
7	001744.jpg	1
11	008003.jpg	1
12	000986.jpg	1

In [0]:
```
predicted_ones_df = results_b[results_b[columns[1]] == 1]
actual_ones_df = imageMap_test[imageMap_test[columns[1]] == 1]

predicted = list(predicted_ones_df['Filenames'])
actual = list(actual_ones_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('Samples of rightly predicted as one\n----------------------')
print(total_match[:5])

print('actual ones: ',len(actual))
print('predicted ones: ',len(predicted))
print('Matched ones(True Positive): ', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of ones: ',prediction_accuracy,'%')

predicted_o_df = results_b[results_b[columns[1]] == 0]
actual_o_df = imageMap_test[imageMap_test[columns[1]] == 0]

predicted = list(predicted_o_df['Filenames'])
actual = list(actual_o_df['Filenames'])

total_match = [i for i in predicted if i in actual]
print('\n\nSamples of rightly predicted as zero\n----------------------')
print(total_match[:5])

print('actual zeros: ',len(actual))
print('predicted zeros: ',len(predicted))
print('Matched zeros (True Negative): ', len(total_match))
prediction_accuracy = (len(total_match) / len(actual))*100
print('prediction accuracy of zeros: ',prediction_accuracy,'%')
```

```
predicted = np.array(results_b[columns[1]])
actual = np.array(imageMap_test[columns[1]])

total_match = np.sum(actual == predicted)
print('\n\nRightly predicted overall\n----------------------')
prediction_accuracy = (total_match / len(actual))*100
print('Number of samples', len(actual))
print('Overall Matches', total_match)
print('Overall prediction accuracy: ',prediction_accuracy,'%')
```

```
Samples of rightly predicted as one
----------------------
['009564.jpg', '000718.jpg', '004703.jpg', '003482.jpg', '008895.jpg']
actual ones:  250
predicted ones:  1565
Matched ones(True Positive):  165
prediction accuracy of ones:  66.0 %


Samples of rightly predicted as zero
----------------------
['003029.jpg', '001600.jpg', '000994.jpg', '000097.jpg', '003378.jpg']
actual zeros:  4702
predicted zeros:  3387
Matched zeros (True Negative):  3302
prediction accuracy of zeros:  70.22543598468737 %


Rightly predicted overall
----------------------
Number of samples 4952
Overall Matches 3467
Overall prediction accuracy:  70.01211631663973 %
```

In [0]:
```
classified_array = np.array(results_b[columns[1]])
results2[columns[1]] = classified_array
results2[:10]
```

Out[0]:

	Filenames	aeroplane	bicycle
0	001966.jpg	0	1
1	003029.jpg	0	0
2	001600.jpg	0	0
3	000994.jpg	1	0
4	000097.jpg	0	0
5	006716.jpg	1	1
6	003378.jpg	0	0
7	001744.jpg	0	1
8	000231.jpg	0	0
9	002703.jpg	1	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c1.csv')
```

Generic Functions

Model 2

In [0]:
```
#Developing a generic function for model 2
def generate_modified_db2(imageMap_trainval, imageMap_test, columns):

    datagen=ImageDataGenerator(rescale=1./255.)
    test_datagen=ImageDataGenerator(rescale=1./255.)

    print('For Training:')
    train_generator=datagen.flow_from_dataframe(
        #dataframe=modifiedDb2[:-100],#modifiedDb[:680], #490
        #dataframe=imageMap_train,
        dataframe=imageMap_trainval[:-200],
        directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
        x_col="Filenames",
        #y_col=training_columns,
        y_col=columns,
        batch_size=32,
        seed=42,
        shuffle=True,
        class_mode="other",
        target_size=(100,100))

    print('For Validation:')
    valid_generator=test_datagen.flow_from_dataframe(
        #dataframe=modifiedDb2[-100:],#modifiedDb[680:],  #490
        #dataframe=imageMap_valid,
        dataframe=imageMap_trainval[-200:],
        directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
        x_col="Filenames",
        #y_col=training_columns,
        y_col=columns,
        batch_size=32,
        seed=42,
        shuffle=True,
        class_mode="other",
        target_size=(100,100))

    print('For Testing:')
    test_generator=test_datagen.flow_from_dataframe(
        dataframe=imageMap_test,
        directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images",
        x_col="Filenames",
        batch_size=1,
        seed=42,
        shuffle=False,
        class_mode=None,
        target_size=(100,100))

    return train_generator, valid_generator, test_generator
```

In [0]:
```
def build_model2():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same',
                     input_shape=(100,100,3)))
    model.add(Activation('relu'))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, (3, 3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(20, activation='sigmoid'))  #Depends on number of classes
    model.compile(optimizers.rmsprop(lr=0.0001, decay=1e-6),loss="binary_crossentropy",metrics=["accuracy"])
    return model
```

MOdel 1

In [0]:
```
#Developing a generic function for all columns: model 1
def generate_modified_db(imageMap_trainval, imageMap_test, columns, coulmn):
    pos_column_db = imageMap_trainval[imageMap_trainval[columns[column]] == 1]
    pos_column_db = pos_column_db.reset_index(drop=True)
    neg_column_db = imageMap_trainval[imageMap_trainval[columns[column]] == 0]
    neg_column_db_shuffled = neg_column_db.sample(frac=1).reset_index(drop=True)
    #neg_column_db_cut = neg_column_db[:len(pos_column_db)*2] #*2 #+50
    neg_column_db_cut = neg_column_db[:2] #*2 #+50

    fraction = int((len(pos_column_db)*2.2)/19)

    for n in range(len(columns)):
        if(columns[n] == columns[column]): print('Avoided :',columns[column])
        else:
          one_neg_sample_db = neg_column_db[neg_column_db[columns[n]] == 1]
          one_neg_sample_db = one_neg_sample_db.sample(frac=1)[:fraction]
          neg_column_db_cut = neg_column_db_cut.append(one_neg_sample_db)
          neg_column_db_cut = neg_column_db_cut.sample(frac=1)
          neg_column_db_cut = neg_column_db_cut.reset_index(drop=True)

    print('Number of positive samples: ',len(pos_column_db))
    print('Number of selected negative samples: ',len(neg_column_db_cut))
    modifiedDb2 = pos_column_db.append(neg_column_db_cut, ignore_index=True)
    modifiedDb2 = modifiedDb2.sample(frac=1).reset_index(drop=True)
    print('Number of merged samples: ',len(modifiedDb2))

    training_columns = [columns[column]]

    datagen=ImageDataGenerator(rescale=1./255.)
    test_datagen=ImageDataGenerator(rescale=1./255.)

    print('For Training:')
    train_generator=datagen.flow_from_dataframe(
        dataframe=modifiedDb2[:-100],#modifiedDb[:680], #490
        #dataframe=imageMap_train,
        directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
        x_col="Filenames",
        y_col=training_columns,
        batch_size=32,
        seed=42,
        shuffle=True,
        class_mode="other",
        target_size=(100,100))

    print('For Validation:')
    valid_generator=test_datagen.flow_from_dataframe(
        dataframe=modifiedDb2[-100:],#modifiedDb[680:],  #490
        #dataframe=imageMap_valid,
        directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Train/Images",
        x_col="Filenames",
        y_col=training_columns,
        batch_size=32,
        seed=42,
        shuffle=True,
        class_mode="other",
        target_size=(100,100))

    print('For Testing:')
    test_generator=test_datagen.flow_from_dataframe(
        dataframe=imageMap_test,
        directory="/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images",
        x_col="Filenames",
        batch_size=1,
        seed=42,
        shuffle=False,
        class_mode=None,
        target_size=(100,100))

    return modifiedDb2, train_generator, valid_generator, test_generator
```

In [0]:
```
def build_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same',
                     input_shape=(100,100,3)))
    model.add(Activation('relu'))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, (3, 3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))  #Depends on number of classes
    model.compile(optimizers.rmsprop(lr=0.0001, decay=1e-6),loss="binary_crossentropy",metrics=["accuracy"])
    return model
```

In [0]:
```
def get_result_db(imageMap_test, test_generator, pred_b, columns, column):
    pred_bool = (pred_b >0.5)

    predictions = pred_bool.astype(int)
    testing_columns = [columns[column]]

    #columns should be the same order of y_col
    results_b = pd.DataFrame(predictions, columns = testing_columns)
    results_b["Filenames"] = test_generator.filenames
    #results["Filenames"] = test_generator.filenames[30]
    ordered_cols = ["Filenames"] + testing_columns
    results_b = results_b[ordered_cols]#To get the same column order
    print('Predicted Results (for ones only)---------------')
    print(results_b[results_b[columns[column]] == 1][:5])

    predicted_ones_df = results_b[results_b[columns[column]] == 1]
    actual_ones_df = imageMap_test[imageMap_test[columns[column]] == 1]

    predicted = list(predicted_ones_df['Filenames'])
    actual = list(actual_ones_df['Filenames'])

    total_match = [i for i in predicted if i in actual]
    print('Samples of rightly predicted as one\n-----------------------')
    print(total_match[:5])

    print('actual ones: ',len(actual))
    print('predicted ones: ',len(predicted))
    print('Matched ones(True Positive): ', len(total_match))
    prediction_accuracy = (len(total_match) / len(actual))*100
    print('prediction accuracy of ones: ',prediction_accuracy,'%')

    predicted_o_df = results_b[results_b[columns[column]] == 0]
    actual_o_df = imageMap_test[imageMap_test[columns[column]] == 0]

    predicted = list(predicted_o_df['Filenames'])
    actual = list(actual_o_df['Filenames'])

    total_match = [i for i in predicted if i in actual]
    print('\n\nSamples of rightly predicted as zero\n-----------------------')
    print(total_match[:5])

    print('actual zeros: ',len(actual))
    print('predicted zeros: ',len(predicted))
    print('Matched zeros (True Negative): ', len(total_match))
    prediction_accuracy = (len(total_match) / len(actual))*100
    print('prediction accuracy of zeros: ',prediction_accuracy,'%')


    predicted = np.array(results_b[columns[column]])
    actual = np.array(imageMap_test[columns[column]])

    total_match = np.sum(actual == predicted)
    print('\n\nRightly predicted overall\n-----------------------')
    prediction_accuracy = (total_match / len(actual))*100
    print('Number of samples', len(actual))
    print('Overall Matches', total_match)
    print('Overall prediction accuracy: ',prediction_accuracy,'%')

    return results_b
```

Training Column 2

In [0]:
```
column = 2
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : bird
Number of positive samples:  333
Number of selected negative samples:  990
Number of merged samples:  1323
For Training:
Found 1191 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	002156.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	
1	000720.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
2	007543.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
3	008914.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
4	006626.jpg	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	
5	009007.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
6	006473.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
7	007847.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	001475.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
9	009422.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_c = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_c = get_result_db(imageMap_test, test_generator, pred_c, columns, column)
```

```
training
-------------------------
Epoch 1/10
37/37 [==============================] - 141s 4s/step - loss: 0.5996 - acc: 0.7340 - val_loss: 0.5817 - val_acc: 0.760
4
Epoch 2/10
37/37 [==============================] - 9s 233ms/step - loss: 0.5634 - acc: 0.7389 - val_loss: 0.5892 - val_acc: 0.76
47
Epoch 3/10
37/37 [==============================] - 7s 186ms/step - loss: 0.5504 - acc: 0.7355 - val_loss: 0.6012 - val_acc: 0.80
88
Epoch 4/10
37/37 [==============================] - 7s 188ms/step - loss: 0.5306 - acc: 0.7469 - val_loss: 0.5653 - val_acc: 0.80
88
Epoch 5/10
37/37 [==============================] - 7s 188ms/step - loss: 0.5009 - acc: 0.7537 - val_loss: 0.5559 - val_acc: 0.83
33
Epoch 6/10
37/37 [==============================] - 7s 187ms/step - loss: 0.5097 - acc: 0.7309 - val_loss: 0.5526 - val_acc: 0.75
00
Epoch 7/10
37/37 [==============================] - 7s 192ms/step - loss: 0.4987 - acc: 0.7541 - val_loss: 0.5243 - val_acc: 0.86
76
Epoch 8/10
37/37 [==============================] - 8s 207ms/step - loss: 0.4765 - acc: 0.7600 - val_loss: 0.6003 - val_acc: 0.69
12
Epoch 9/10
37/37 [==============================] - 7s 191ms/step - loss: 0.4746 - acc: 0.7630 - val_loss: 0.5351 - val_acc: 0.82
29
Epoch 10/10
37/37 [==============================] - 7s 186ms/step - loss: 0.4549 - acc: 0.7588 - val_loss: 0.5249 - val_acc: 0.79
41

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  bird
0   001019.jpg     1
5   008839.jpg     1
16  006090.jpg     1
28  007253.jpg     1
35  004845.jpg     1
Samples of rightly predicted as one
-----------------------
['009158.jpg', '004350.jpg', '000773.jpg', '009232.jpg', '000383.jpg']
actual ones:  289
predicted ones:  921
Matched ones(True Positive):  139
prediction accuracy of ones:  48.09688581314879 %


Samples of rightly predicted as zero
-----------------------
['001639.jpg', '000371.jpg', '007032.jpg', '002185.jpg', '001210.jpg']
actual zeros:  4663
predicted zeros:  4031
Matched zeros (True Negative):  3881
prediction accuracy of zeros:  83.2296804632211 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 4020
Overall prediction accuracy:  81.17932148626818 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
results2 = pd.read_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c1.csv')
classified_array = np.array(results_c[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird
0	0	001966.jpg	0	1	1
1	1	003029.jpg	0	0	0
2	2	001600.jpg	0	0	0
3	3	000994.jpg	1	0	0
4	4	000097.jpg	0	0	0
5	5	006716.jpg	1	1	1
6	6	003378.jpg	0	0	0
7	7	001744.jpg	0	1	0
8	8	000231.jpg	0	0	0
9	9	002703.jpg	1	0	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c2.csv')
```

Training Column 3

In [0]:
```
column = 3
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : boat
Number of positive samples:  188
Number of selected negative samples:  933
Number of merged samples:  1121
For Training:
Found 993 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	003863.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	
1	002916.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
2	003422.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	
3	003797.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	
4	007956.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	
5	009526.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	
6	005047.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
7	005714.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	
8	001680.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
9	001642.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_d = model.predict_generator(test_generator,
                                 steps=STEP_SIZE_TEST,
                                 #steps = 100,
                                 verbose=1)

print('\nResult showcasing\n-------------------------')
results_d = get_result_db(imageMap_test, test_generator, pred_d, columns, column)
```

```
training
-------------------------
Epoch 1/10
31/31 [==============================] - 108s 3s/step - loss: 0.5252 - acc: 0.8175 - val_loss: 0.5228 - val_acc: 0.854
2
Epoch 2/10
31/31 [==============================] - 6s 178ms/step - loss: 0.4741 - acc: 0.7982 - val_loss: 0.4708 - val_acc: 0.79
41
Epoch 3/10
31/31 [==============================] - 6s 184ms/step - loss: 0.4003 - acc: 0.8396 - val_loss: 0.6133 - val_acc: 0.66
18
Epoch 4/10
31/31 [==============================] - 6s 187ms/step - loss: 0.3849 - acc: 0.8375 - val_loss: 0.6541 - val_acc: 0.60
29
Epoch 5/10
31/31 [==============================] - 6s 187ms/step - loss: 0.3442 - acc: 0.8587 - val_loss: 0.6881 - val_acc: 0.52
08
Epoch 6/10
31/31 [==============================] - 6s 180ms/step - loss: 0.3511 - acc: 0.8507 - val_loss: 0.4930 - val_acc: 0.85
29
Epoch 7/10
31/31 [==============================] - 6s 181ms/step - loss: 0.3370 - acc: 0.8476 - val_loss: 0.4094 - val_acc: 0.83
82
Epoch 8/10
31/31 [==============================] - 6s 196ms/step - loss: 0.3377 - acc: 0.8577 - val_loss: 0.5463 - val_acc: 0.73
53
Epoch 9/10
31/31 [==============================] - 7s 219ms/step - loss: 0.3219 - acc: 0.8587 - val_loss: 0.4316 - val_acc: 0.87
50
Epoch 10/10
31/31 [==============================] - 6s 188ms/step - loss: 0.3475 - acc: 0.8567 - val_loss: 0.4517 - val_acc: 0.76
47

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  boat
2   000371.jpg     1
13  009856.jpg     1
30  000004.jpg     1
33  005572.jpg     1
35  004845.jpg     1
Samples of rightly predicted as one
-----------------------
['000371.jpg', '007233.jpg', '001474.jpg', '000792.jpg', '006977.jpg']
actual ones:  176
predicted ones:  900
Matched ones(True Positive):  111
prediction accuracy of ones:  63.06818181818182 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '007032.jpg', '002185.jpg', '008839.jpg']
actual zeros:  4776
predicted zeros:  4052
Matched zeros (True Negative):  3987
prediction accuracy of zeros:  83.47989949748744 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 4098
Overall prediction accuracy:  82.75444264943457 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
classified_array = np.array(results_d[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat
0	0	001966.jpg	0	1	1	0
1	1	003029.jpg	0	0	0	0
2	2	001600.jpg	0	0	0	1
3	3	000994.jpg	1	0	0	0
4	4	000097.jpg	0	0	0	0
5	5	006716.jpg	1	1	1	0
6	6	003378.jpg	0	0	0	0
7	7	001744.jpg	0	1	0	0
8	8	000231.jpg	0	0	0	0
9	9	002703.jpg	1	0	0	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c3.csv')
```

Training Column 4

In [0]:
```
column = 4
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : bottle
Number of positive samples:  262
Number of selected negative samples:  933
Number of merged samples:  1195
For Training:
Found 1062 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	008702.jpg	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	
1	004655.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	
2	009405.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
3	006084.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
4	001699.jpg	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	
5	000929.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	
6	000855.jpg	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	
7	005563.jpg	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	
8	003555.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	
9	003865.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_e = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_e = get_result_db(imageMap_test, test_generator, pred_e, columns, column)
```

```
training
-------------------------
Epoch 1/20
33/33 [==============================] - 49s 1s/step - loss: 0.5392 - acc: 0.7652 - val_loss: 0.5421 - val_acc: 0.7812
Epoch 2/20
33/33 [==============================] - 6s 191ms/step - loss: 0.5176 - acc: 0.7731 - val_loss: 0.5208 - val_acc: 0.73
53
Epoch 3/20
33/33 [==============================] - 6s 182ms/step - loss: 0.4978 - acc: 0.7724 - val_loss: 0.4768 - val_acc: 0.77
94
Epoch 4/20
33/33 [==============================] - 6s 183ms/step - loss: 0.4962 - acc: 0.7686 - val_loss: 0.5046 - val_acc: 0.77
94
Epoch 5/20
33/33 [==============================] - 6s 186ms/step - loss: 0.4595 - acc: 0.7932 - val_loss: 0.5628 - val_acc: 0.71
88
Epoch 6/20
33/33 [==============================] - 6s 184ms/step - loss: 0.4489 - acc: 0.7942 - val_loss: 0.5060 - val_acc: 0.77
94
Epoch 7/20
33/33 [==============================] - 6s 182ms/step - loss: 0.4829 - acc: 0.7820 - val_loss: 0.5065 - val_acc: 0.76
47
Epoch 8/20
33/33 [==============================] - 6s 183ms/step - loss: 0.4574 - acc: 0.7970 - val_loss: 0.5146 - val_acc: 0.75
00
Epoch 9/20
33/33 [==============================] - 6s 186ms/step - loss: 0.4307 - acc: 0.8048 - val_loss: 0.4861 - val_acc: 0.76
04
Epoch 10/20
33/33 [==============================] - 7s 204ms/step - loss: 0.4369 - acc: 0.8084 - val_loss: 0.4854 - val_acc: 0.77
94
Epoch 11/20
33/33 [==============================] - 7s 200ms/step - loss: 0.4364 - acc: 0.8169 - val_loss: 0.4860 - val_acc: 0.75
00
Epoch 12/20
33/33 [==============================] - 6s 185ms/step - loss: 0.4177 - acc: 0.8103 - val_loss: 0.4802 - val_acc: 0.73
53
Epoch 13/20
33/33 [==============================] - 6s 188ms/step - loss: 0.4199 - acc: 0.8171 - val_loss: 0.5020 - val_acc: 0.75
00
Epoch 14/20
33/33 [==============================] - 6s 187ms/step - loss: 0.4021 - acc: 0.8218 - val_loss: 0.4583 - val_acc: 0.80
88
Epoch 15/20
33/33 [==============================] - 6s 184ms/step - loss: 0.4086 - acc: 0.8198 - val_loss: 0.5175 - val_acc: 0.67
65
Epoch 16/20
33/33 [==============================] - 6s 185ms/step - loss: 0.3843 - acc: 0.8379 - val_loss: 0.4977 - val_acc: 0.75
00
Epoch 17/20
33/33 [==============================] - 6s 188ms/step - loss: 0.3743 - acc: 0.8317 - val_loss: 0.5460 - val_acc: 0.66
67
Epoch 18/20
33/33 [==============================] - 6s 182ms/step - loss: 0.3670 - acc: 0.8577 - val_loss: 0.4684 - val_acc: 0.79
41
Epoch 19/20
33/33 [==============================] - 6s 185ms/step - loss: 0.3327 - acc: 0.8597 - val_loss: 0.5351 - val_acc: 0.72
06
Epoch 20/20
33/33 [==============================] - 6s 184ms/step - loss: 0.3450 - acc: 0.8434 - val_loss: 0.5284 - val_acc: 0.64
71

testing
-------------------------
4952/4952 [==============================] - 74s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  bottle
0   001019.jpg       1
10  005155.jpg       1
14  008889.jpg       1
17  004226.jpg       1
21  009492.jpg       1
Samples of rightly predicted as one
-----------------------
['009075.jpg', '002207.jpg', '009798.jpg', '003802.jpg', '007744.jpg']
actual ones:  240
predicted ones:  1472
Matched ones(True Positive):  119
prediction accuracy of ones:  49.583333333333336 %


Samples of rightly predicted as zero
-----------------------
['001639.jpg', '000371.jpg', '007032.jpg', '002185.jpg', '008839.jpg']
actual zeros:  4712
predicted zeros:  3480
Matched zeros (True Negative):  3359
prediction accuracy of zeros:  71.28607809847199 %


Rightly predicted overall
-----------------------
```

```
Number of samples 4952
Overall Matches 3478
Overall prediction accuracy:  70.23424878836833 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
classified_array = np.array(results_e[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle
0	0	001966.jpg	0	1	1	0	1
1	1	003029.jpg	0	0	0	0	0
2	2	001600.jpg	0	0	0	1	0
3	3	000994.jpg	1	0	0	0	0
4	4	000097.jpg	0	0	0	0	0
5	5	006716.jpg	1	1	1	0	0
6	6	003378.jpg	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0
8	8	000231.jpg	0	0	0	0	0
9	9	002703.jpg	1	0	0	0	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c4.csv')
```

Training Column 5

In [0]:
```
column = 5
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : bus
Number of positive samples:  197
Number of selected negative samples:  705
Number of merged samples:  902
For Training:
Found 786 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	009136.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	
1	009420.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	
2	009279.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	
3	008633.jpg	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
4	009244.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
5	007565.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
6	003065.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	
7	007685.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	003599.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	001009.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=30 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_f = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_f = get_result_db(imageMap_test, test_generator, pred_f, columns, column)
```

```
training
-------------------------
Epoch 1/30
24/24 [==============================] - 6s 239ms/step - loss: 0.5945 - acc: 0.7629 - val_loss: 0.6001 - val_acc: 0.72
06
Epoch 2/30
24/24 [==============================] - 4s 181ms/step - loss: 0.5114 - acc: 0.7847 - val_loss: 0.5539 - val_acc: 0.75
00
Epoch 3/30
24/24 [==============================] - 5s 189ms/step - loss: 0.4911 - acc: 0.7826 - val_loss: 0.6274 - val_acc: 0.70
59
Epoch 4/30
24/24 [==============================] - 5s 188ms/step - loss: 0.4641 - acc: 0.7778 - val_loss: 0.5970 - val_acc: 0.72
06
Epoch 5/30
24/24 [==============================] - 4s 185ms/step - loss: 0.4437 - acc: 0.7919 - val_loss: 0.5179 - val_acc: 0.80
88
Epoch 6/30
24/24 [==============================] - 5s 190ms/step - loss: 0.4403 - acc: 0.8085 - val_loss: 0.5678 - val_acc: 0.73
96
Epoch 7/30
24/24 [==============================] - 4s 186ms/step - loss: 0.4476 - acc: 0.7785 - val_loss: 0.5676 - val_acc: 0.73
53
Epoch 8/30
24/24 [==============================] - 4s 185ms/step - loss: 0.4131 - acc: 0.8117 - val_loss: 0.4989 - val_acc: 0.73
53
Epoch 9/30
24/24 [==============================] - 4s 186ms/step - loss: 0.4015 - acc: 0.8078 - val_loss: 0.5825 - val_acc: 0.76
47
Epoch 10/30
24/24 [==============================] - 5s 189ms/step - loss: 0.4156 - acc: 0.8029 - val_loss: 0.5452 - val_acc: 0.71
88
Epoch 11/30
24/24 [==============================] - 4s 187ms/step - loss: 0.3759 - acc: 0.8228 - val_loss: 0.5626 - val_acc: 0.70
59
Epoch 12/30
24/24 [==============================] - 4s 187ms/step - loss: 0.3983 - acc: 0.8081 - val_loss: 0.5180 - val_acc: 0.76
47
Epoch 13/30
24/24 [==============================] - 5s 189ms/step - loss: 0.3688 - acc: 0.8332 - val_loss: 0.5655 - val_acc: 0.72
06
Epoch 14/30
24/24 [==============================] - 5s 197ms/step - loss: 0.3813 - acc: 0.8134 - val_loss: 0.5830 - val_acc: 0.67
71
Epoch 15/30
24/24 [==============================] - 5s 211ms/step - loss: 0.3409 - acc: 0.8472 - val_loss: 0.6524 - val_acc: 0.64
71
Epoch 16/30
24/24 [==============================] - 5s 204ms/step - loss: 0.3629 - acc: 0.8407 - val_loss: 0.4667 - val_acc: 0.76
47
Epoch 17/30
24/24 [==============================] - 5s 188ms/step - loss: 0.3310 - acc: 0.8541 - val_loss: 0.5876 - val_acc: 0.70
59
Epoch 18/30
24/24 [==============================] - 5s 191ms/step - loss: 0.3279 - acc: 0.8635 - val_loss: 0.6051 - val_acc: 0.70
83
Epoch 19/30
24/24 [==============================] - 4s 186ms/step - loss: 0.3312 - acc: 0.8573 - val_loss: 0.5934 - val_acc: 0.73
53
Epoch 20/30
24/24 [==============================] - 5s 188ms/step - loss: 0.3371 - acc: 0.8472 - val_loss: 0.5517 - val_acc: 0.69
12
Epoch 21/30
24/24 [==============================] - 4s 185ms/step - loss: 0.3113 - acc: 0.8557 - val_loss: 0.6017 - val_acc: 0.72
06
Epoch 22/30
24/24 [==============================] - 5s 191ms/step - loss: 0.2706 - acc: 0.8824 - val_loss: 0.5403 - val_acc: 0.71
88
Epoch 23/30
24/24 [==============================] - 4s 186ms/step - loss: 0.2899 - acc: 0.8805 - val_loss: 0.6637 - val_acc: 0.67
65
Epoch 24/30
24/24 [==============================] - 5s 189ms/step - loss: 0.2757 - acc: 0.8844 - val_loss: 0.7394 - val_acc: 0.67
65
Epoch 25/30
24/24 [==============================] - 5s 191ms/step - loss: 0.2850 - acc: 0.8831 - val_loss: 0.6496 - val_acc: 0.70
59
Epoch 26/30
24/24 [==============================] - 5s 191ms/step - loss: 0.2466 - acc: 0.9003 - val_loss: 0.6529 - val_acc: 0.70
83
Epoch 27/30
24/24 [==============================] - 4s 186ms/step - loss: 0.2410 - acc: 0.8990 - val_loss: 0.6992 - val_acc: 0.69
12
Epoch 28/30
24/24 [==============================] - 5s 189ms/step - loss: 0.2478 - acc: 0.9062 - val_loss: 0.6496 - val_acc: 0.70
59
Epoch 29/30
24/24 [==============================] - 4s 186ms/step - loss: 0.2293 - acc: 0.8958 - val_loss: 0.7767 - val_acc: 0.57
35
Epoch 30/30
24/24 [==============================] - 5s 191ms/step - loss: 0.2067 - acc: 0.9215 - val_loss: 0.7680 - val_acc: 0.71
88

testing
-------------------------
```

```
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  bus
4   002185.jpg    1
12  005927.jpg    1
13  009856.jpg    1
19  003286.jpg    1
31  002148.jpg    1
Samples of rightly predicted as one
----------------------
['005265.jpg', '006992.jpg', '004144.jpg', '009510.jpg', '007858.jpg']
actual ones:  183
predicted ones:  530
Matched ones(True Positive):  84
prediction accuracy of ones:  45.90163934426229 %


Samples of rightly predicted as zero
----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '007032.jpg', '008839.jpg']
actual zeros:  4769
predicted zeros:  4422
Matched zeros (True Negative):  4323
prediction accuracy of zeros:  90.64793457747956 %


Rightly predicted overall
----------------------
Number of samples 4952
Overall Matches 4407
Overall prediction accuracy:  88.99434571890146 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_f[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
-------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus
0	0	001966.jpg	0	1	1	0	1	0
1	1	003029.jpg	0	0	0	0	0	0
2	2	001600.jpg	0	0	0	1	0	0
3	3	000994.jpg	1	0	0	0	0	0
4	4	000097.jpg	0	0	0	0	0	1
5	5	006716.jpg	1	1	1	0	0	0
6	6	003378.jpg	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0
8	8	000231.jpg	0	0	0	0	0	0
9	9	002703.jpg	1	0	0	0	0	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c5.csv')
```

Training Column 6

In [0]:
```
column = 6
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : car
Number of positive samples:  761
Number of selected negative samples:  2652
Number of merged samples:  3413
For Training:
Found 3058 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	006241.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	
1	000443.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	
2	003629.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
3	000854.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	
4	006210.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
5	001409.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
6	007212.jpg	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	
7	000871.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
8	007419.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
9	000050.jpg	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_g = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_g = get_result_db(imageMap_test, test_generator, pred_g, columns, column)
```

```
training
-------------------------
Epoch 1/20
95/95 [==============================] - 198s 2s/step - loss: 0.5547 - acc: 0.7559 - val_loss: 0.5853 - val_acc: 0.739
6
Epoch 2/20
95/95 [==============================] - 17s 184ms/step - loss: 0.5128 - acc: 0.7669 - val_loss: 0.5144 - val_acc: 0.7
500
Epoch 3/20
95/95 [==============================] - 17s 184ms/step - loss: 0.4998 - acc: 0.7664 - val_loss: 0.5789 - val_acc: 0.7
206
Epoch 4/20
95/95 [==============================] - 18s 185ms/step - loss: 0.4792 - acc: 0.7690 - val_loss: 0.4860 - val_acc: 0.7
941
Epoch 5/20
95/95 [==============================] - 19s 198ms/step - loss: 0.4725 - acc: 0.7729 - val_loss: 0.5104 - val_acc: 0.7
083
Epoch 6/20
95/95 [==============================] - 18s 185ms/step - loss: 0.4608 - acc: 0.7806 - val_loss: 0.5928 - val_acc: 0.6
765
Epoch 7/20
95/95 [==============================] - 18s 185ms/step - loss: 0.4582 - acc: 0.7773 - val_loss: 0.4643 - val_acc: 0.7
941
Epoch 8/20
95/95 [==============================] - 18s 185ms/step - loss: 0.4465 - acc: 0.7838 - val_loss: 0.4951 - val_acc: 0.7
059
Epoch 9/20
95/95 [==============================] - 18s 185ms/step - loss: 0.4310 - acc: 0.8027 - val_loss: 0.4812 - val_acc: 0.7
396
Epoch 10/20
95/95 [==============================] - 19s 197ms/step - loss: 0.4271 - acc: 0.8052 - val_loss: 0.5669 - val_acc: 0.6
471
Epoch 11/20
95/95 [==============================] - 19s 195ms/step - loss: 0.4127 - acc: 0.8077 - val_loss: 0.4662 - val_acc: 0.7
206
Epoch 12/20
95/95 [==============================] - 19s 201ms/step - loss: 0.4194 - acc: 0.8014 - val_loss: 0.4636 - val_acc: 0.7
500
Epoch 13/20
95/95 [==============================] - 18s 186ms/step - loss: 0.3950 - acc: 0.8163 - val_loss: 0.4952 - val_acc: 0.7
292
Epoch 14/20
95/95 [==============================] - 19s 199ms/step - loss: 0.3990 - acc: 0.8100 - val_loss: 0.4267 - val_acc: 0.7
794
Epoch 15/20
95/95 [==============================] - 18s 186ms/step - loss: 0.3828 - acc: 0.8190 - val_loss: 0.4765 - val_acc: 0.7
794
Epoch 16/20
95/95 [==============================] - 18s 185ms/step - loss: 0.3781 - acc: 0.8297 - val_loss: 0.5458 - val_acc: 0.6
912
Epoch 17/20
95/95 [==============================] - 18s 186ms/step - loss: 0.3775 - acc: 0.8228 - val_loss: 0.4826 - val_acc: 0.7
917
Epoch 18/20
95/95 [==============================] - 18s 188ms/step - loss: 0.3719 - acc: 0.8244 - val_loss: 0.4987 - val_acc: 0.7
500
Epoch 19/20
95/95 [==============================] - 19s 195ms/step - loss: 0.3520 - acc: 0.8467 - val_loss: 0.4971 - val_acc: 0.7
206
Epoch 20/20
95/95 [==============================] - 18s 185ms/step - loss: 0.3635 - acc: 0.8356 - val_loss: 0.5045 - val_acc: 0.7
647

testing
-------------------------
4952/4952 [==============================] - 72s 14ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  car
3   007032.jpg    1
4   002185.jpg    1
12  005927.jpg    1
21  009492.jpg    1
27  001335.jpg    1
Samples of rightly predicted as one
-----------------------
['002185.jpg', '009492.jpg', '001335.jpg', '001619.jpg', '001883.jpg']
actual ones:  775
predicted ones:  1178
Matched ones(True Positive):  482
prediction accuracy of ones:  62.193548387096776 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '008839.jpg', '001210.jpg']
actual zeros:  4177
predicted zeros:  3774
Matched zeros (True Negative):  3481
prediction accuracy of zeros:  83.33732343787406 %


Rightly predicted overall
```

```
----------------------
Number of samples 4952
Overall Matches 3963
Overall prediction accuracy:  80.02827140549273 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_g[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car
0	0	001966.jpg	0	1	1	0	1	0	0
1	1	003029.jpg	0	0	0	0	0	0	0
2	2	001600.jpg	0	0	0	1	0	0	0
3	3	000994.jpg	1	0	0	0	0	0	1
4	4	000097.jpg	0	0	0	0	0	1	1
5	5	006716.jpg	1	1	1	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0
8	8	000231.jpg	0	0	0	0	0	0	0
9	9	002703.jpg	1	0	0	0	0	0	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c6.csv')
```

Training Column 7

In [0]:
```
column = 7
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : cat
Number of positive samples:  344
Number of selected negative samples:  1237
Number of merged samples:  1581
For Training:
Found 1413 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	003468.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
1	000118.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
2	004066.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	002963.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
4	001982.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
5	009726.jpg	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	
6	006988.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
7	001782.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	002276.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	009527.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_h = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_h = get_result_db(imageMap_test, test_generator, pred_h, columns, column)
```

```
training
-------------------------
Epoch 1/20
44/44 [==============================] - 32s 723ms/step - loss: 0.5315 - acc: 0.7678 - val_loss: 0.4762 - val_acc: 0.8
021
Epoch 2/20
44/44 [==============================] - 8s 188ms/step - loss: 0.4738 - acc: 0.7790 - val_loss: 0.5760 - val_acc: 0.83
82
Epoch 3/20
44/44 [==============================] - 8s 181ms/step - loss: 0.4779 - acc: 0.7673 - val_loss: 0.4270 - val_acc: 0.85
29
Epoch 4/20
44/44 [==============================] - 8s 183ms/step - loss: 0.4552 - acc: 0.7784 - val_loss: 0.4655 - val_acc: 0.85
29
Epoch 5/20
44/44 [==============================] - 8s 186ms/step - loss: 0.4467 - acc: 0.7895 - val_loss: 0.4205 - val_acc: 0.86
46
Epoch 6/20
44/44 [==============================] - 8s 182ms/step - loss: 0.4344 - acc: 0.7939 - val_loss: 0.4446 - val_acc: 0.83
82
Epoch 7/20
44/44 [==============================] - 8s 183ms/step - loss: 0.4186 - acc: 0.8153 - val_loss: 0.4642 - val_acc: 0.79
41
Epoch 8/20
44/44 [==============================] - 8s 181ms/step - loss: 0.4267 - acc: 0.7987 - val_loss: 0.3385 - val_acc: 0.86
76
Epoch 9/20
44/44 [==============================] - 8s 185ms/step - loss: 0.4193 - acc: 0.8021 - val_loss: 0.3739 - val_acc: 0.85
42
Epoch 10/20
44/44 [==============================] - 8s 184ms/step - loss: 0.3937 - acc: 0.8288 - val_loss: 0.4728 - val_acc: 0.80
88
Epoch 11/20
44/44 [==============================] - 9s 200ms/step - loss: 0.3947 - acc: 0.8224 - val_loss: 0.2596 - val_acc: 0.91
18
Epoch 12/20
44/44 [==============================] - 9s 194ms/step - loss: 0.3631 - acc: 0.8293 - val_loss: 0.3505 - val_acc: 0.80
88
Epoch 13/20
44/44 [==============================] - 8s 186ms/step - loss: 0.3822 - acc: 0.8224 - val_loss: 0.3174 - val_acc: 0.87
50
Epoch 14/20
44/44 [==============================] - 8s 184ms/step - loss: 0.3735 - acc: 0.8323 - val_loss: 0.3287 - val_acc: 0.88
24
Epoch 15/20
44/44 [==============================] - 8s 184ms/step - loss: 0.3495 - acc: 0.8472 - val_loss: 0.3753 - val_acc: 0.77
94
Epoch 16/20
44/44 [==============================] - 8s 184ms/step - loss: 0.3464 - acc: 0.8399 - val_loss: 0.2722 - val_acc: 0.89
71
Epoch 17/20
44/44 [==============================] - 8s 186ms/step - loss: 0.3403 - acc: 0.8536 - val_loss: 0.3143 - val_acc: 0.87
50
Epoch 18/20
44/44 [==============================] - 8s 183ms/step - loss: 0.3293 - acc: 0.8556 - val_loss: 0.3905 - val_acc: 0.83
82
Epoch 19/20
44/44 [==============================] - 8s 183ms/step - loss: 0.3017 - acc: 0.8625 - val_loss: 0.3782 - val_acc: 0.85
29
Epoch 20/20
44/44 [==============================] - 8s 182ms/step - loss: 0.3118 - acc: 0.8627 - val_loss: 0.3634 - val_acc: 0.83
82

testing
-------------------------
4952/4952 [==============================] - 72s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  cat
0   001019.jpg    1
11  004599.jpg    1
14  008889.jpg    1
16  006090.jpg    1
17  004226.jpg    1
Samples of rightly predicted as one
-----------------------
['001019.jpg', '006090.jpg', '005309.jpg', '004845.jpg', '002740.jpg']
actual ones:  332
predicted ones:  1358
Matched ones(True Positive):  218
prediction accuracy of ones:  65.66265060240963 %


Samples of rightly predicted as zero
-----------------------
['001639.jpg', '000371.jpg', '007032.jpg', '002185.jpg', '008839.jpg']
actual zeros:  4620
predicted zeros:  3594
Matched zeros (True Negative):  3480
prediction accuracy of zeros:  75.32467532467533 %


Rightly predicted overall
```

```
----------------------
Number of samples 4952
Overall Matches 3698
Overall prediction accuracy:  74.67689822294022 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_h[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat
0	0	001966.jpg	0	1	1	0	1	0	0	1
1	1	003029.jpg	0	0	0	0	0	0	0	0
2	2	001600.jpg	0	0	0	1	0	0	0	0
3	3	000994.jpg	1	0	0	0	0	0	1	0
4	4	000097.jpg	0	0	0	0	0	1	1	0
5	5	006716.jpg	1	1	1	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0
8	8	000231.jpg	0	0	0	0	0	0	0	0
9	9	002703.jpg	1	0	0	0	0	0	0	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c7.csv')
```

Training Column 8

In [0]:
```
column = 8
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : chair
Number of positive samples:  572
Number of selected negative samples:  2015
Number of merged samples:  2587
For Training:
Found 2359 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	009375.jpg	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	
1	006847.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
2	002221.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	
3	006046.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	003362.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
5	004954.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
6	005964.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
7	004950.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
8	005752.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
9	001945.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_i = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_i = get_result_db(imageMap_test, test_generator, pred_i, columns, column)
```

```
training
-------------------------
Epoch 1/20
73/73 [==============================] - 43s 595ms/step - loss: 0.5483 - acc: 0.7671 - val_loss: 0.4803 - val_acc: 0.7
708
Epoch 2/20
73/73 [==============================] - 15s 202ms/step - loss: 0.5158 - acc: 0.7678 - val_loss: 0.4532 - val_acc: 0.7
941
Epoch 3/20
73/73 [==============================] - 14s 186ms/step - loss: 0.5004 - acc: 0.7687 - val_loss: 0.6085 - val_acc: 0.7
353
Epoch 4/20
73/73 [==============================] - 13s 185ms/step - loss: 0.4949 - acc: 0.7665 - val_loss: 0.4388 - val_acc: 0.7
794
Epoch 5/20
73/73 [==============================] - 14s 187ms/step - loss: 0.4874 - acc: 0.7713 - val_loss: 0.4542 - val_acc: 0.7
708
Epoch 6/20
73/73 [==============================] - 14s 186ms/step - loss: 0.4797 - acc: 0.7702 - val_loss: 0.4313 - val_acc: 0.7
794
Epoch 7/20
73/73 [==============================] - 14s 189ms/step - loss: 0.4677 - acc: 0.7701 - val_loss: 0.4516 - val_acc: 0.8
088
Epoch 8/20
73/73 [==============================] - 15s 199ms/step - loss: 0.4619 - acc: 0.7772 - val_loss: 0.4746 - val_acc: 0.7
647
Epoch 9/20
73/73 [==============================] - 14s 187ms/step - loss: 0.4445 - acc: 0.7769 - val_loss: 0.4444 - val_acc: 0.7
812
Epoch 10/20
73/73 [==============================] - 14s 186ms/step - loss: 0.4480 - acc: 0.7838 - val_loss: 0.4440 - val_acc: 0.7
941
Epoch 11/20
73/73 [==============================] - 14s 189ms/step - loss: 0.4356 - acc: 0.7856 - val_loss: 0.4433 - val_acc: 0.8
088
Epoch 12/20
73/73 [==============================] - 14s 186ms/step - loss: 0.4284 - acc: 0.7888 - val_loss: 0.4697 - val_acc: 0.7
500
Epoch 13/20
73/73 [==============================] - 15s 209ms/step - loss: 0.4256 - acc: 0.8032 - val_loss: 0.4274 - val_acc: 0.8
125
Epoch 14/20
73/73 [==============================] - 14s 193ms/step - loss: 0.4024 - acc: 0.8096 - val_loss: 0.3945 - val_acc: 0.8
235
Epoch 15/20
73/73 [==============================] - 14s 185ms/step - loss: 0.4079 - acc: 0.8059 - val_loss: 0.4288 - val_acc: 0.7
794
Epoch 16/20
73/73 [==============================] - 14s 188ms/step - loss: 0.3985 - acc: 0.8021 - val_loss: 0.5710 - val_acc: 0.7
353
Epoch 17/20
73/73 [==============================] - 14s 186ms/step - loss: 0.3932 - acc: 0.8103 - val_loss: 0.4590 - val_acc: 0.7
812
Epoch 18/20
73/73 [==============================] - 14s 185ms/step - loss: 0.3882 - acc: 0.8210 - val_loss: 0.4692 - val_acc: 0.7
500
Epoch 19/20
73/73 [==============================] - 14s 197ms/step - loss: 0.3716 - acc: 0.8321 - val_loss: 0.3844 - val_acc: 0.7
794
Epoch 20/20
73/73 [==============================] - 14s 190ms/step - loss: 0.3612 - acc: 0.8280 - val_loss: 0.5369 - val_acc: 0.6
618

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  chair
7   009739.jpg      1
8   006003.jpg      1
13  009856.jpg      1
15  007835.jpg      1
17  004226.jpg      1
Samples of rightly predicted as one
-----------------------
['007835.jpg', '002788.jpg', '009075.jpg', '009632.jpg', '005040.jpg']
actual ones:  545
predicted ones:  1398
Matched ones(True Positive):  350
prediction accuracy of ones:  64.22018348623854 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '007032.jpg', '002185.jpg']
actual zeros:  4407
predicted zeros:  3554
Matched zeros (True Negative):  3359
prediction accuracy of zeros:  76.21965055593374 %


Rightly predicted overall
```

```
----------------------
Number of samples 4952
Overall Matches 3709
Overall prediction accuracy:  74.89903069466882 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_i[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair
0	0	001966.jpg	0	1	1	0	1	0	0	1	0
1	1	003029.jpg	0	0	0	0	0	0	0	0	0
2	2	001600.jpg	0	0	0	1	0	0	0	0	0
3	3	000994.jpg	1	0	0	0	0	0	1	0	0
4	4	000097.jpg	0	0	0	0	0	1	1	0	0
5	5	006716.jpg	1	1	1	0	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0	1
8	8	000231.jpg	0	0	0	0	0	0	0	0	1
9	9	002703.jpg	1	0	0	0	0	0	0	0	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c8.csv')
```

Training Column 9

In [0]:
```
column = 9
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : cow
Number of positive samples:  146
Number of selected negative samples:  515
Number of merged samples:  661
For Training:
Found 550 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	004742.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1	1	0	
1	009878.jpg	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	
2	004436.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	008226.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	002589.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	
5	008086.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
6	009905.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	002868.jpg	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
8	004708.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
9	005841.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_j = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_j = get_result_db(imageMap_test, test_generator, pred_j, columns, column)
```

```
training
-------------------------
Epoch 1/20
17/17 [==============================] - 7s 425ms/step - loss: 0.6797 - acc: 0.7353 - val_loss: 0.4959 - val_acc: 0.81
25
Epoch 2/20
17/17 [==============================] - 3s 171ms/step - loss: 0.5523 - acc: 0.7768 - val_loss: 0.5098 - val_acc: 0.83
82
Epoch 3/20
17/17 [==============================] - 3s 184ms/step - loss: 0.5663 - acc: 0.7656 - val_loss: 0.5387 - val_acc: 0.80
88
Epoch 4/20
17/17 [==============================] - 3s 185ms/step - loss: 0.5677 - acc: 0.7545 - val_loss: 0.5529 - val_acc: 0.82
35
Epoch 5/20
17/17 [==============================] - 3s 187ms/step - loss: 0.5462 - acc: 0.7619 - val_loss: 0.5276 - val_acc: 0.81
25
Epoch 6/20
17/17 [==============================] - 3s 183ms/step - loss: 0.5160 - acc: 0.7768 - val_loss: 0.5495 - val_acc: 0.79
41
Epoch 7/20
17/17 [==============================] - 3s 183ms/step - loss: 0.5260 - acc: 0.7545 - val_loss: 0.5635 - val_acc: 0.88
24
Epoch 8/20
17/17 [==============================] - 3s 184ms/step - loss: 0.5072 - acc: 0.7750 - val_loss: 0.5664 - val_acc: 0.79
41
Epoch 9/20
17/17 [==============================] - 3s 189ms/step - loss: 0.5144 - acc: 0.7601 - val_loss: 0.5608 - val_acc: 0.82
29
Epoch 10/20
17/17 [==============================] - 3s 184ms/step - loss: 0.4805 - acc: 0.7676 - val_loss: 0.5344 - val_acc: 0.82
35
Epoch 11/20
17/17 [==============================] - 3s 182ms/step - loss: 0.4666 - acc: 0.7825 - val_loss: 0.5265 - val_acc: 0.83
82
Epoch 12/20
17/17 [==============================] - 3s 181ms/step - loss: 0.4541 - acc: 0.7750 - val_loss: 0.5120 - val_acc: 0.80
88
Epoch 13/20
17/17 [==============================] - 3s 187ms/step - loss: 0.4594 - acc: 0.7713 - val_loss: 0.5217 - val_acc: 0.82
29
Epoch 14/20
17/17 [==============================] - 3s 181ms/step - loss: 0.4252 - acc: 0.8100 - val_loss: 0.4805 - val_acc: 0.77
94
Epoch 15/20
17/17 [==============================] - 3s 187ms/step - loss: 0.4161 - acc: 0.8101 - val_loss: 0.4378 - val_acc: 0.85
29
Epoch 16/20
17/17 [==============================] - 3s 186ms/step - loss: 0.4513 - acc: 0.7747 - val_loss: 0.5173 - val_acc: 0.73
53
Epoch 17/20
17/17 [==============================] - 4s 218ms/step - loss: 0.4235 - acc: 0.8061 - val_loss: 0.4698 - val_acc: 0.80
21
Epoch 18/20
17/17 [==============================] - 4s 219ms/step - loss: 0.4054 - acc: 0.7877 - val_loss: 0.5527 - val_acc: 0.69
12
Epoch 19/20
17/17 [==============================] - 4s 231ms/step - loss: 0.4038 - acc: 0.8070 - val_loss: 0.5032 - val_acc: 0.79
41
Epoch 20/20
17/17 [==============================] - 4s 220ms/step - loss: 0.3841 - acc: 0.8303 - val_loss: 0.4787 - val_acc: 0.80
88

testing
-------------------------
4952/4952 [==============================] - 71s 14ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  cow
2   000371.jpg    1
7   009739.jpg    1
9   005722.jpg    1
11  004599.jpg    1
12  005927.jpg    1
Samples of rightly predicted as one
-----------------------
['004022.jpg', '004181.jpg', '009264.jpg', '004332.jpg', '003201.jpg']
actual ones:  127
predicted ones:  963
Matched ones(True Positive):  77
prediction accuracy of ones:  60.629921259842526 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '007032.jpg', '002185.jpg', '008839.jpg']
actual zeros:  4825
predicted zeros:  3989
Matched zeros (True Negative):  3939
prediction accuracy of zeros:  81.63730569948187 %


Rightly predicted overall
```

```
----------------------
Number of samples 4952
Overall Matches 4016
Overall prediction accuracy:  81.09854604200322 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_j[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c9.csv')
```

Training Column 10

In [0]:
```
column = 10
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : diningtable
Number of positive samples:  263
Number of selected negative samples:  933
Number of merged samples:  1196
For Training:
Found 1066 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	006437.jpg	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
1	003863.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	
2	001490.jpg	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	
3	004452.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	004834.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
5	001263.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	
6	006699.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	
7	005481.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
8	000531.jpg	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	
9	000754.jpg	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_k = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_k = get_result_db(imageMap_test, test_generator, pred_k, columns, column)
```

```
training
-------------------------
Epoch 1/10
33/33 [==============================] - 9s 259ms/step - loss: 0.5460 - acc: 0.7638 - val_loss: 0.5884 - val_acc: 0.72
06
Epoch 2/10
33/33 [==============================] - 6s 186ms/step - loss: 0.5132 - acc: 0.7666 - val_loss: 0.5498 - val_acc: 0.79
17
Epoch 3/10
33/33 [==============================] - 6s 180ms/step - loss: 0.4941 - acc: 0.7580 - val_loss: 0.6444 - val_acc: 0.67
65
Epoch 4/10
33/33 [==============================] - 6s 182ms/step - loss: 0.4400 - acc: 0.7970 - val_loss: 0.5332 - val_acc: 0.70
59
Epoch 5/10
33/33 [==============================] - 6s 182ms/step - loss: 0.4963 - acc: 0.7591 - val_loss: 0.5981 - val_acc: 0.72
06
Epoch 6/10
33/33 [==============================] - 6s 185ms/step - loss: 0.4516 - acc: 0.7809 - val_loss: 0.5803 - val_acc: 0.71
88
Epoch 7/10
33/33 [==============================] - 6s 181ms/step - loss: 0.4662 - acc: 0.7769 - val_loss: 0.5740 - val_acc: 0.75
00
Epoch 8/10
33/33 [==============================] - 6s 181ms/step - loss: 0.4358 - acc: 0.7856 - val_loss: 0.6321 - val_acc: 0.63
24
Epoch 9/10
33/33 [==============================] - 6s 181ms/step - loss: 0.4371 - acc: 0.7970 - val_loss: 0.5730 - val_acc: 0.73
53
Epoch 10/10
33/33 [==============================] - 6s 185ms/step - loss: 0.4307 - acc: 0.7830 - val_loss: 0.5860 - val_acc: 0.72
92

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  diningtable
0   001019.jpg            1
1   001639.jpg            1
8   006003.jpg            1
10  005155.jpg            1
14  008889.jpg            1
Samples of rightly predicted as one
-----------------------
['000006.jpg', '009514.jpg', '007393.jpg', '006546.jpg', '008113.jpg']
actual ones:  247
predicted ones:  1119
Matched ones(True Positive):  146
prediction accuracy of ones:  59.10931174089069 %


Samples of rightly predicted as zero
-----------------------
['000371.jpg', '007032.jpg', '002185.jpg', '008839.jpg', '001210.jpg']
actual zeros:  4705
predicted zeros:  3833
Matched zeros (True Negative):  3732
prediction accuracy of zeros:  79.31987247608927 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 3878
Overall prediction accuracy:  78.31179321486267 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
classified_array = np.array(results_k[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c10.csv')
```

Training Column 11

In [0]:
```
column = 11
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : dog
Number of positive samples:  430
Number of selected negative samples:  990
Number of merged samples:  1420
For Training:
Found 1289 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	003403.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
1	008036.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
2	000772.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
3	008509.jpg	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	
4	008840.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
5	004548.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
6	008346.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
7	001555.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
8	001630.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
9	004686.jpg	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_1 = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_1 = get_result_db(imageMap_test, test_generator, pred_1, columns, column)
```

```
training
-------------------------
Epoch 1/20
40/40 [==============================] - 11s 279ms/step - loss: 0.6723 - acc: 0.6617 - val_loss: 0.5752 - val_acc: 0.7
188
Epoch 2/20
40/40 [==============================] - 7s 172ms/step - loss: 0.6060 - acc: 0.6815 - val_loss: 0.6144 - val_acc: 0.70
59
Epoch 3/20
40/40 [==============================] - 7s 178ms/step - loss: 0.5884 - acc: 0.6952 - val_loss: 0.6216 - val_acc: 0.82
35
Epoch 4/20
40/40 [==============================] - 7s 179ms/step - loss: 0.5860 - acc: 0.6874 - val_loss: 0.6154 - val_acc: 0.77
94
Epoch 5/20
40/40 [==============================] - 8s 204ms/step - loss: 0.5701 - acc: 0.7003 - val_loss: 0.5766 - val_acc: 0.81
25
Epoch 6/20
40/40 [==============================] - 8s 212ms/step - loss: 0.5640 - acc: 0.6909 - val_loss: 0.6211 - val_acc: 0.69
12
Epoch 7/20
40/40 [==============================] - 7s 180ms/step - loss: 0.5574 - acc: 0.6960 - val_loss: 0.6260 - val_acc: 0.73
53
Epoch 8/20
40/40 [==============================] - 7s 178ms/step - loss: 0.5483 - acc: 0.7163 - val_loss: 0.5681 - val_acc: 0.75
00
Epoch 9/20
40/40 [==============================] - 7s 181ms/step - loss: 0.5438 - acc: 0.7069 - val_loss: 0.5285 - val_acc: 0.77
08
Epoch 10/20
40/40 [==============================] - 7s 179ms/step - loss: 0.5243 - acc: 0.7304 - val_loss: 0.5013 - val_acc: 0.82
35
Epoch 11/20
40/40 [==============================] - 7s 180ms/step - loss: 0.5366 - acc: 0.7128 - val_loss: 0.6021 - val_acc: 0.69
12
Epoch 12/20
40/40 [==============================] - 8s 195ms/step - loss: 0.5121 - acc: 0.7452 - val_loss: 0.4792 - val_acc: 0.79
41
Epoch 13/20
40/40 [==============================] - 9s 214ms/step - loss: 0.5246 - acc: 0.7378 - val_loss: 0.4822 - val_acc: 0.79
17
Epoch 14/20
40/40 [==============================] - 7s 178ms/step - loss: 0.4922 - acc: 0.7448 - val_loss: 0.5304 - val_acc: 0.79
41
Epoch 15/20
40/40 [==============================] - 7s 185ms/step - loss: 0.5123 - acc: 0.7405 - val_loss: 0.4197 - val_acc: 0.83
82
Epoch 16/20
40/40 [==============================] - 8s 209ms/step - loss: 0.4821 - acc: 0.7718 - val_loss: 0.4890 - val_acc: 0.76
47
Epoch 17/20
40/40 [==============================] - 7s 183ms/step - loss: 0.4833 - acc: 0.7565 - val_loss: 0.4962 - val_acc: 0.79
17
Epoch 18/20
40/40 [==============================] - 7s 179ms/step - loss: 0.4665 - acc: 0.7655 - val_loss: 0.5023 - val_acc: 0.76
47
Epoch 19/20
40/40 [==============================] - 7s 180ms/step - loss: 0.4623 - acc: 0.7687 - val_loss: 0.4616 - val_acc: 0.80
88
Epoch 20/20
40/40 [==============================] - 7s 179ms/step - loss: 0.4394 - acc: 0.8019 - val_loss: 0.4747 - val_acc: 0.77
94

testing
-------------------------
4952/4952 [==============================] - 71s 14ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  dog
0   001019.jpg    1
7   009739.jpg    1
9   005722.jpg    1
11  004599.jpg    1
14  008889.jpg    1
Samples of rightly predicted as one
----------------------
['008889.jpg', '003224.jpg', '006487.jpg', '008656.jpg', '005050.jpg']
actual ones:  433
predicted ones:  1242
Matched ones(True Positive):  218
prediction accuracy of ones:  50.34642032332564 %


Samples of rightly predicted as zero
----------------------
['000371.jpg', '007032.jpg', '002185.jpg', '008839.jpg', '001210.jpg']
actual zeros:  4519
predicted zeros:  3710
Matched zeros (True Negative):  3495
prediction accuracy of zeros:  77.3401194954636 %


Rightly predicted overall
```

```
----------------------
Number of samples 4952
Overall Matches 3713
Overall prediction accuracy:  74.97980613893377 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_1[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c11.csv')
```

Training Column 12

In [0]:
```
column = 12
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : horse
Number of positive samples:  294
Number of selected negative samples:  667
Number of merged samples:  961
For Training:
Found 845 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	004849.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	006628.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	
2	004015.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
3	008867.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
4	006833.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
5	002648.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
6	007637.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
7	009168.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
8	002366.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
9	004535.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_m = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_m = get_result_db(imageMap_test, test_generator, pred_m, columns, column)
```

```
training
-------------------------
Epoch 1/20
26/26 [==============================] - 8s 309ms/step - loss: 0.6513 - acc: 0.6647 - val_loss: 0.6327 - val_acc: 0.68
75
Epoch 2/20
26/26 [==============================] - 4s 171ms/step - loss: 0.6115 - acc: 0.6732 - val_loss: 0.6185 - val_acc: 0.75
00
Epoch 3/20
26/26 [==============================] - 5s 180ms/step - loss: 0.6051 - acc: 0.6855 - val_loss: 0.6498 - val_acc: 0.72
06
Epoch 4/20
26/26 [==============================] - 5s 181ms/step - loss: 0.5658 - acc: 0.7153 - val_loss: 0.6854 - val_acc: 0.55
88
Epoch 5/20
26/26 [==============================] - 5s 184ms/step - loss: 0.5356 - acc: 0.7209 - val_loss: 0.5668 - val_acc: 0.71
88
Epoch 6/20
26/26 [==============================] - 5s 186ms/step - loss: 0.5071 - acc: 0.7452 - val_loss: 0.5485 - val_acc: 0.76
47
Epoch 7/20
26/26 [==============================] - 5s 210ms/step - loss: 0.5045 - acc: 0.7664 - val_loss: 0.6708 - val_acc: 0.61
76
Epoch 8/20
26/26 [==============================] - 5s 199ms/step - loss: 0.5069 - acc: 0.7512 - val_loss: 0.4959 - val_acc: 0.77
94
Epoch 9/20
26/26 [==============================] - 5s 186ms/step - loss: 0.4390 - acc: 0.7950 - val_loss: 0.6635 - val_acc: 0.58
33
Epoch 10/20
26/26 [==============================] - 5s 182ms/step - loss: 0.4869 - acc: 0.7919 - val_loss: 0.5826 - val_acc: 0.66
18
Epoch 11/20
26/26 [==============================] - 5s 182ms/step - loss: 0.4407 - acc: 0.8094 - val_loss: 0.5651 - val_acc: 0.69
12
Epoch 12/20
26/26 [==============================] - 5s 182ms/step - loss: 0.4338 - acc: 0.8046 - val_loss: 0.5889 - val_acc: 0.60
29
Epoch 13/20
26/26 [==============================] - 5s 185ms/step - loss: 0.4022 - acc: 0.8368 - val_loss: 0.5589 - val_acc: 0.69
79
Epoch 14/20
26/26 [==============================] - 5s 180ms/step - loss: 0.4143 - acc: 0.8114 - val_loss: 0.5826 - val_acc: 0.70
59
Epoch 15/20
26/26 [==============================] - 5s 181ms/step - loss: 0.3910 - acc: 0.8262 - val_loss: 0.6300 - val_acc: 0.63
24
Epoch 16/20
26/26 [==============================] - 5s 181ms/step - loss: 0.3626 - acc: 0.8354 - val_loss: 0.5159 - val_acc: 0.73
53
Epoch 17/20
26/26 [==============================] - 5s 186ms/step - loss: 0.3464 - acc: 0.8277 - val_loss: 0.5845 - val_acc: 0.66
67
Epoch 18/20
26/26 [==============================] - 5s 181ms/step - loss: 0.3327 - acc: 0.8491 - val_loss: 0.6402 - val_acc: 0.67
65
Epoch 19/20
26/26 [==============================] - 5s 181ms/step - loss: 0.3287 - acc: 0.8472 - val_loss: 0.4291 - val_acc: 0.75
00
Epoch 20/20
26/26 [==============================] - 5s 180ms/step - loss: 0.2993 - acc: 0.8660 - val_loss: 0.6387 - val_acc: 0.64
71

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  horse
4   002185.jpg      1
9   005722.jpg      1
11  004599.jpg      1
12  005927.jpg      1
24  008325.jpg      1
Samples of rightly predicted as one
-----------------------
['005722.jpg', '007253.jpg', '000356.jpg', '006002.jpg', '002421.jpg']
actual ones:  279
predicted ones:  1091
Matched ones(True Positive):  203
prediction accuracy of ones:  72.75985663082437 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '007032.jpg', '008839.jpg']
actual zeros:  4673
predicted zeros:  3861
Matched zeros (True Negative):  3785
prediction accuracy of zeros:  80.99721806120266 %


Rightly predicted overall
```

```
----------------------
Number of samples 4952
Overall Matches 3988
Overall prediction accuracy:  80.53311793214863 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_m[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1	0
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0	0
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0	0
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0	1
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1	0
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1	1

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c12.csv')
```

Training Column 13

In [0]:
```
column = 13
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : motorbike
Number of positive samples:  249
Number of selected negative samples:  572
Number of merged samples:  821
For Training:
Found 715 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	009306.jpg	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	
1	009698.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
2	002958.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
3	008150.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
4	001073.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
5	004158.jpg	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	
6	003078.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	
7	004987.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
8	002585.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
9	007322.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_n = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_n = get_result_db(imageMap_test, test_generator, pred_n, columns, column)
```

```
training
-------------------------
Epoch 1/20
22/22 [==============================] - 7s 323ms/step - loss: 0.7344 - acc: 0.6420 - val_loss: 0.6703 - val_acc: 0.73
96
Epoch 2/20
22/22 [==============================] - 4s 169ms/step - loss: 0.6313 - acc: 0.6881 - val_loss: 0.5469 - val_acc: 0.82
35
Epoch 3/20
22/22 [==============================] - 4s 185ms/step - loss: 0.6124 - acc: 0.6952 - val_loss: 0.5642 - val_acc: 0.73
53
Epoch 4/20
22/22 [==============================] - 4s 186ms/step - loss: 0.5765 - acc: 0.7089 - val_loss: 0.5066 - val_acc: 0.76
47
Epoch 5/20
22/22 [==============================] - 4s 190ms/step - loss: 0.5418 - acc: 0.7400 - val_loss: 0.4777 - val_acc: 0.77
08
Epoch 6/20
22/22 [==============================] - 4s 184ms/step - loss: 0.5177 - acc: 0.7552 - val_loss: 0.5616 - val_acc: 0.70
59
Epoch 7/20
22/22 [==============================] - 4s 183ms/step - loss: 0.5265 - acc: 0.7661 - val_loss: 0.4750 - val_acc: 0.82
35
Epoch 8/20
22/22 [==============================] - 4s 195ms/step - loss: 0.4570 - acc: 0.8113 - val_loss: 0.4536 - val_acc: 0.82
35
Epoch 9/20
22/22 [==============================] - 5s 215ms/step - loss: 0.4450 - acc: 0.8068 - val_loss: 0.4676 - val_acc: 0.79
17
Epoch 10/20
22/22 [==============================] - 4s 203ms/step - loss: 0.4487 - acc: 0.8068 - val_loss: 0.5266 - val_acc: 0.69
12
Epoch 11/20
22/22 [==============================] - 4s 183ms/step - loss: 0.4888 - acc: 0.7765 - val_loss: 0.4936 - val_acc: 0.77
94
Epoch 12/20
22/22 [==============================] - 4s 184ms/step - loss: 0.4149 - acc: 0.8443 - val_loss: 0.5130 - val_acc: 0.72
06
Epoch 13/20
22/22 [==============================] - 4s 187ms/step - loss: 0.4306 - acc: 0.8213 - val_loss: 0.4519 - val_acc: 0.80
21
Epoch 14/20
22/22 [==============================] - 4s 183ms/step - loss: 0.4400 - acc: 0.8035 - val_loss: 0.4966 - val_acc: 0.73
53
Epoch 15/20
22/22 [==============================] - 4s 183ms/step - loss: 0.4008 - acc: 0.8270 - val_loss: 0.5123 - val_acc: 0.79
41
Epoch 16/20
22/22 [==============================] - 4s 183ms/step - loss: 0.4014 - acc: 0.8298 - val_loss: 0.6081 - val_acc: 0.61
76
Epoch 17/20
22/22 [==============================] - 4s 186ms/step - loss: 0.3592 - acc: 0.8403 - val_loss: 0.4952 - val_acc: 0.76
04
Epoch 18/20
22/22 [==============================] - 4s 182ms/step - loss: 0.3584 - acc: 0.8637 - val_loss: 0.5351 - val_acc: 0.73
53
Epoch 19/20
22/22 [==============================] - 4s 182ms/step - loss: 0.3693 - acc: 0.8424 - val_loss: 0.6489 - val_acc: 0.64
71
Epoch 20/20
22/22 [==============================] - 4s 182ms/step - loss: 0.3456 - acc: 0.8602 - val_loss: 0.3590 - val_acc: 0.86
76

testing
-------------------------
4952/4952 [==============================] - 72s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  motorbike
4   002185.jpg          1
10  005155.jpg          1
27  001335.jpg          1
47  001883.jpg          1
54  001648.jpg          1
Samples of rightly predicted as one
----------------------
['002185.jpg', '004641.jpg', '006823.jpg', '001031.jpg', '002806.jpg']
actual ones:  233
predicted ones:  567
Matched ones(True Positive):  123
prediction accuracy of ones:  52.78969957081545 %


Samples of rightly predicted as zero
----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '007032.jpg', '008839.jpg']
actual zeros:  4719
predicted zeros:  4385
Matched zeros (True Negative):  4275
prediction accuracy of zeros:  90.59122695486332 %


Rightly predicted overall
```

```
----------------------
Number of samples 4952
Overall Matches 4398
Overall prediction accuracy:  88.81260096930534 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_n[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1	0	0
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0	0	0
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0	0	0
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0	1	1
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1	0	0
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1	1	0

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c13.csv')
```

Training Column 14

In [0]:
```
column = 14
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : person
Number of positive samples:  2095
Number of selected negative samples:  2970
Number of merged samples:  5065
For Training:
Found 2501 images.
For Validation:
Found 2510 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	000374.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
1	002368.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	
2	009389.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	004140.jpg	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
4	005507.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
5	009209.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
6	002278.jpg	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
7	006784.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
8	004347.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	002598.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=30 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_p = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_p = get_result_db(imageMap_test, test_generator, pred_p, columns, column)
```

```
training
-------------------------
Epoch 1/30
78/78 [==============================] - 29s 365ms/step - loss: 0.6942 - acc: 0.5661 - val_loss: 0.6677 - val_acc: 0.5
974
Epoch 2/30
78/78 [==============================] - 28s 360ms/step - loss: 0.6658 - acc: 0.5894 - val_loss: 0.6599 - val_acc: 0.6
126
Epoch 3/30
78/78 [==============================] - 25s 324ms/step - loss: 0.6537 - acc: 0.6004 - val_loss: 0.6531 - val_acc: 0.6
219
Epoch 4/30
78/78 [==============================] - 25s 320ms/step - loss: 0.6414 - acc: 0.6282 - val_loss: 0.6451 - val_acc: 0.6
283
Epoch 5/30
78/78 [==============================] - 25s 326ms/step - loss: 0.6356 - acc: 0.6330 - val_loss: 0.6564 - val_acc: 0.6
158
Epoch 6/30
78/78 [==============================] - 26s 329ms/step - loss: 0.6330 - acc: 0.6438 - val_loss: 0.6623 - val_acc: 0.5
993
Epoch 7/30
78/78 [==============================] - 25s 317ms/step - loss: 0.6230 - acc: 0.6543 - val_loss: 0.6424 - val_acc: 0.6
429
Epoch 8/30
78/78 [==============================] - 25s 318ms/step - loss: 0.6068 - acc: 0.6710 - val_loss: 0.6320 - val_acc: 0.6
449
Epoch 9/30
78/78 [==============================] - 27s 346ms/step - loss: 0.6020 - acc: 0.6801 - val_loss: 0.6379 - val_acc: 0.6
437
Epoch 10/30
78/78 [==============================] - 25s 319ms/step - loss: 0.5897 - acc: 0.6943 - val_loss: 0.6388 - val_acc: 0.6
437
Epoch 11/30
78/78 [==============================] - 25s 318ms/step - loss: 0.5781 - acc: 0.7005 - val_loss: 0.6345 - val_acc: 0.6
469
Epoch 12/30
78/78 [==============================] - 26s 338ms/step - loss: 0.5596 - acc: 0.7120 - val_loss: 0.6233 - val_acc: 0.6
562
Epoch 13/30
78/78 [==============================] - 25s 317ms/step - loss: 0.5373 - acc: 0.7338 - val_loss: 0.6257 - val_acc: 0.6
481
Epoch 14/30
78/78 [==============================] - 27s 344ms/step - loss: 0.5279 - acc: 0.7436 - val_loss: 0.6231 - val_acc: 0.6
630
Epoch 15/30
78/78 [==============================] - 26s 336ms/step - loss: 0.5250 - acc: 0.7430 - val_loss: 0.6383 - val_acc: 0.6
404
Epoch 16/30
78/78 [==============================] - 25s 318ms/step - loss: 0.5100 - acc: 0.7500 - val_loss: 0.6348 - val_acc: 0.6
533
Epoch 17/30
78/78 [==============================] - 25s 317ms/step - loss: 0.4835 - acc: 0.7707 - val_loss: 0.6444 - val_acc: 0.6
404
Epoch 18/30
78/78 [==============================] - 26s 336ms/step - loss: 0.4605 - acc: 0.7847 - val_loss: 0.6407 - val_acc: 0.6
517
Epoch 19/30
78/78 [==============================] - 25s 318ms/step - loss: 0.4469 - acc: 0.7955 - val_loss: 0.6674 - val_acc: 0.6
493
Epoch 20/30
78/78 [==============================] - 25s 316ms/step - loss: 0.4265 - acc: 0.8059 - val_loss: 0.6713 - val_acc: 0.6
533
Epoch 21/30
78/78 [==============================] - 26s 337ms/step - loss: 0.4025 - acc: 0.8256 - val_loss: 0.6834 - val_acc: 0.6
469
Epoch 22/30
78/78 [==============================] - 25s 320ms/step - loss: 0.4026 - acc: 0.8250 - val_loss: 0.6918 - val_acc: 0.6
348
Epoch 23/30
78/78 [==============================] - 25s 318ms/step - loss: 0.3636 - acc: 0.8449 - val_loss: 0.7131 - val_acc: 0.6
348
Epoch 24/30
78/78 [==============================] - 25s 325ms/step - loss: 0.3471 - acc: 0.8432 - val_loss: 0.6936 - val_acc: 0.6
554
Epoch 25/30
78/78 [==============================] - 26s 328ms/step - loss: 0.3268 - acc: 0.8575 - val_loss: 0.7776 - val_acc: 0.6
312
Epoch 26/30
78/78 [==============================] - 26s 335ms/step - loss: 0.3005 - acc: 0.8762 - val_loss: 0.8139 - val_acc: 0.6
437
Epoch 27/30
78/78 [==============================] - 25s 326ms/step - loss: 0.2738 - acc: 0.8902 - val_loss: 0.8070 - val_acc: 0.6
481
Epoch 28/30
78/78 [==============================] - 26s 334ms/step - loss: 0.2743 - acc: 0.8901 - val_loss: 0.8050 - val_acc: 0.6
275
Epoch 29/30
78/78 [==============================] - 25s 318ms/step - loss: 0.2543 - acc: 0.9096 - val_loss: 0.8910 - val_acc: 0.6
154
Epoch 30/30
78/78 [==============================] - 25s 317ms/step - loss: 0.2222 - acc: 0.9205 - val_loss: 0.8632 - val_acc: 0.6
211

testing
-------------------------
```

```
4952/4952 [==============================] - 71s 14ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
    Filenames  person
1  001639.jpg       1
3  007032.jpg       1
4  002185.jpg       1
7  009739.jpg       1
8  006003.jpg       1
Samples of rightly predicted as one
-----------------------
['007032.jpg', '002185.jpg', '009739.jpg', '005722.jpg', '004599.jpg']
actual ones:  2097
predicted ones:  2159
Matched ones(True Positive):  1204
prediction accuracy of ones:  57.415355269432524 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '000371.jpg', '005155.jpg', '005927.jpg', '009856.jpg']
actual zeros:  2855
predicted zeros:  2793
Matched zeros (True Negative):  1900
prediction accuracy of zeros:  66.54991243432575 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 3104
Overall prediction accuracy:  62.68174474959613 %
```

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=50 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_o = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_o = get_result_db(imageMap_test, test_generator, pred_o, columns, column)
```

```
training
------------------------
Epoch 1/50
139/139 [==============================] - 29s 211ms/step - loss: 0.7018 - acc: 0.5436 - val_loss: 0.6458 - val_acc:
0.6250
Epoch 2/50
139/139 [==============================] - 25s 182ms/step - loss: 0.6587 - acc: 0.6087 - val_loss: 0.6448 - val_acc:
0.6765
Epoch 3/50
139/139 [==============================] - 26s 184ms/step - loss: 0.6450 - acc: 0.6298 - val_loss: 0.5951 - val_acc:
0.6618
Epoch 4/50
139/139 [==============================] - 27s 194ms/step - loss: 0.6313 - acc: 0.6481 - val_loss: 0.6329 - val_acc:
0.6618
Epoch 5/50
139/139 [==============================] - 26s 188ms/step - loss: 0.6128 - acc: 0.6668 - val_loss: 0.6053 - val_acc:
0.6979
Epoch 6/50
139/139 [==============================] - 26s 191ms/step - loss: 0.6023 - acc: 0.6789 - val_loss: 0.6073 - val_acc:
0.7059
Epoch 7/50
139/139 [==============================] - 27s 191ms/step - loss: 0.5807 - acc: 0.6977 - val_loss: 0.5829 - val_acc:
0.6618
Epoch 8/50
139/139 [==============================] - 26s 186ms/step - loss: 0.5597 - acc: 0.7164 - val_loss: 0.6060 - val_acc:
0.6912
Epoch 9/50
139/139 [==============================] - 28s 200ms/step - loss: 0.5406 - acc: 0.7321 - val_loss: 0.5787 - val_acc:
0.7083
Epoch 10/50
139/139 [==============================] - 27s 191ms/step - loss: 0.5232 - acc: 0.7463 - val_loss: 0.5729 - val_acc:
0.7206
Epoch 11/50
139/139 [==============================] - 26s 185ms/step - loss: 0.5064 - acc: 0.7540 - val_loss: 0.5652 - val_acc:
0.6765
Epoch 12/50
139/139 [==============================] - 25s 183ms/step - loss: 0.4868 - acc: 0.7721 - val_loss: 0.5512 - val_acc:
0.7647
Epoch 13/50
139/139 [==============================] - 26s 188ms/step - loss: 0.4474 - acc: 0.7980 - val_loss: 0.5320 - val_acc:
0.7500
Epoch 14/50
139/139 [==============================] - 26s 187ms/step - loss: 0.4355 - acc: 0.8017 - val_loss: 0.5993 - val_acc:
0.6618
Epoch 15/50
139/139 [==============================] - 25s 182ms/step - loss: 0.4122 - acc: 0.8228 - val_loss: 0.4998 - val_acc:
0.7794
Epoch 16/50
139/139 [==============================] - 26s 184ms/step - loss: 0.3869 - acc: 0.8244 - val_loss: 0.5256 - val_acc:
0.7353
Epoch 17/50
139/139 [==============================] - 27s 197ms/step - loss: 0.3582 - acc: 0.8460 - val_loss: 0.5726 - val_acc:
0.7188
Epoch 18/50
139/139 [==============================] - 25s 181ms/step - loss: 0.3463 - acc: 0.8536 - val_loss: 0.4595 - val_acc:
0.8088
Epoch 19/50
139/139 [==============================] - 25s 181ms/step - loss: 0.3151 - acc: 0.8676 - val_loss: 0.6150 - val_acc:
0.6765
Epoch 20/50
139/139 [==============================] - 27s 191ms/step - loss: 0.2936 - acc: 0.8825 - val_loss: 0.5779 - val_acc:
0.7941
Epoch 21/50
139/139 [==============================] - 27s 195ms/step - loss: 0.2693 - acc: 0.8880 - val_loss: 0.5796 - val_acc:
0.7188
Epoch 22/50
139/139 [==============================] - 25s 182ms/step - loss: 0.2409 - acc: 0.9074 - val_loss: 0.6955 - val_acc:
0.6912
Epoch 23/50
139/139 [==============================] - 27s 192ms/step - loss: 0.2265 - acc: 0.9114 - val_loss: 0.6762 - val_acc:
0.6765
Epoch 24/50
139/139 [==============================] - 26s 185ms/step - loss: 0.2058 - acc: 0.9247 - val_loss: 0.5354 - val_acc:
0.7941
Epoch 25/50
139/139 [==============================] - 26s 184ms/step - loss: 0.1903 - acc: 0.9300 - val_loss: 0.6028 - val_acc:
0.7396
Epoch 26/50
139/139 [==============================] - 27s 191ms/step - loss: 0.1690 - acc: 0.9374 - val_loss: 0.6768 - val_acc:
0.7647
Epoch 27/50
139/139 [==============================] - 25s 182ms/step - loss: 0.1525 - acc: 0.9429 - val_loss: 0.9169 - val_acc:
0.6765
Epoch 28/50
139/139 [==============================] - 25s 182ms/step - loss: 0.1350 - acc: 0.9533 - val_loss: 0.5482 - val_acc:
0.7647
Epoch 29/50
139/139 [==============================] - 27s 195ms/step - loss: 0.1090 - acc: 0.9614 - val_loss: 0.8317 - val_acc:
0.7188
Epoch 30/50
139/139 [==============================] - 25s 182ms/step - loss: 0.1094 - acc: 0.9631 - val_loss: 0.4635 - val_acc:
0.8088
Epoch 31/50
139/139 [==============================] - 25s 183ms/step - loss: 0.0901 - acc: 0.9726 - val_loss: 1.0197 - val_acc:
0.7059
```

```
Epoch 32/50
139/139 [==============================] - 27s 194ms/step - loss: 0.0827 - acc: 0.9714 - val_loss: 0.7409 - val_acc:
0.7794
Epoch 33/50
139/139 [==============================] - 27s 196ms/step - loss: 0.0758 - acc: 0.9752 - val_loss: 0.7741 - val_acc:
0.7396
Epoch 34/50
139/139 [==============================] - 25s 183ms/step - loss: 0.0694 - acc: 0.9761 - val_loss: 0.6389 - val_acc:
0.7647
Epoch 35/50
139/139 [==============================] - 27s 193ms/step - loss: 0.0594 - acc: 0.9805 - val_loss: 0.9854 - val_acc:
0.7647
Epoch 36/50
139/139 [==============================] - 25s 181ms/step - loss: 0.0572 - acc: 0.9834 - val_loss: 1.0915 - val_acc:
0.7059
Epoch 37/50
139/139 [==============================] - 25s 182ms/step - loss: 0.0473 - acc: 0.9843 - val_loss: 0.9931 - val_acc:
0.7917
Epoch 38/50
139/139 [==============================] - 27s 193ms/step - loss: 0.0475 - acc: 0.9864 - val_loss: 0.8366 - val_acc:
0.7353
Epoch 39/50
139/139 [==============================] - 26s 186ms/step - loss: 0.0410 - acc: 0.9861 - val_loss: 0.7199 - val_acc:
0.7794
Epoch 40/50
139/139 [==============================] - 25s 183ms/step - loss: 0.0451 - acc: 0.9852 - val_loss: 0.9848 - val_acc:
0.7059
Epoch 41/50
139/139 [==============================] - 28s 198ms/step - loss: 0.0310 - acc: 0.9912 - val_loss: 1.2886 - val_acc:
0.6979
Epoch 42/50
139/139 [==============================] - 25s 181ms/step - loss: 0.0379 - acc: 0.9874 - val_loss: 0.8829 - val_acc:
0.7500
Epoch 43/50
139/139 [==============================] - 25s 182ms/step - loss: 0.0378 - acc: 0.9885 - val_loss: 0.9294 - val_acc:
0.7941
Epoch 44/50
139/139 [==============================] - 26s 188ms/step - loss: 0.0314 - acc: 0.9899 - val_loss: 1.2030 - val_acc:
0.6912
Epoch 45/50
139/139 [==============================] - 28s 200ms/step - loss: 0.0312 - acc: 0.9890 - val_loss: 1.2609 - val_acc:
0.7083
Epoch 46/50
139/139 [==============================] - 26s 184ms/step - loss: 0.0315 - acc: 0.9915 - val_loss: 1.1734 - val_acc:
0.7500
Epoch 47/50
139/139 [==============================] - 26s 187ms/step - loss: 0.0275 - acc: 0.9910 - val_loss: 1.0020 - val_acc:
0.7794
Epoch 48/50
139/139 [==============================] - 26s 187ms/step - loss: 0.0323 - acc: 0.9897 - val_loss: 1.1420 - val_acc:
0.7500
Epoch 49/50
139/139 [==============================] - 25s 182ms/step - loss: 0.0230 - acc: 0.9924 - val_loss: 1.1226 - val_acc:
0.7500
Epoch 50/50
139/139 [==============================] - 26s 184ms/step - loss: 0.0262 - acc: 0.9899 - val_loss: 1.0289 - val_acc:
0.7794

testing
-------------------------
4952/4952 [==============================] - 71s 14ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  person
3   007032.jpg       1
7   009739.jpg       1
8   006003.jpg       1
9   005722.jpg       1
10  005155.jpg       1
Samples of rightly predicted as one
-----------------------
['007032.jpg', '009739.jpg', '005722.jpg', '000955.jpg', '009075.jpg']
actual ones:  2097
predicted ones:  1980
Matched ones(True Positive):  1137
prediction accuracy of ones:  54.220314735336196 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '005927.jpg', '009856.jpg']
actual zeros:  2855
predicted zeros:  2972
Matched zeros (True Negative):  2012
prediction accuracy of zeros:  70.47285464098073 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 3149
Overall prediction accuracy:  63.59046849757674 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
classified_array = np.array(results_o[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c14.csv')
```

Training Column 15

In [0]:
```
column = 15
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : pottedplant
Number of positive samples:  273
Number of selected negative samples:  591
Number of merged samples:  864
For Training:
Found 757 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	000337.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
1	004528.jpg	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
2	007715.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
3	004714.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	
4	002525.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	000484.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
6	007673.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	
7	000381.jpg	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	
8	006575.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
9	006411.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=30 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_q = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_q = get_result_db(imageMap_test, test_generator, pred_q, columns, column)
```

```
training
------------------------
Epoch 1/30
22/22 [==============================] - 7s 328ms/step - loss: 0.6877 - acc: 0.6264 - val_loss: 0.6364 - val_acc: 0.72
92
Epoch 2/30
22/22 [==============================] - 4s 173ms/step - loss: 0.6456 - acc: 0.6573 - val_loss: 0.6644 - val_acc: 0.67
65
Epoch 3/30
22/22 [==============================] - 4s 189ms/step - loss: 0.6304 - acc: 0.6771 - val_loss: 0.6375 - val_acc: 0.73
53
Epoch 4/30
22/22 [==============================] - 4s 189ms/step - loss: 0.6173 - acc: 0.6757 - val_loss: 0.5966 - val_acc: 0.77
94
Epoch 5/30
22/22 [==============================] - 4s 194ms/step - loss: 0.6078 - acc: 0.6684 - val_loss: 0.6103 - val_acc: 0.71
88
Epoch 6/30
22/22 [==============================] - 4s 188ms/step - loss: 0.6329 - acc: 0.6428 - val_loss: 0.6535 - val_acc: 0.66
18
Epoch 7/30
22/22 [==============================] - 4s 188ms/step - loss: 0.5794 - acc: 0.6941 - val_loss: 0.6282 - val_acc: 0.73
53
Epoch 8/30
22/22 [==============================] - 4s 191ms/step - loss: 0.5956 - acc: 0.6844 - val_loss: 0.5904 - val_acc: 0.76
47
Epoch 9/30
22/22 [==============================] - 4s 193ms/step - loss: 0.5724 - acc: 0.7084 - val_loss: 0.5457 - val_acc: 0.75
00
Epoch 10/30
22/22 [==============================] - 4s 186ms/step - loss: 0.5776 - acc: 0.6956 - val_loss: 0.6536 - val_acc: 0.60
29
Epoch 11/30
22/22 [==============================] - 4s 188ms/step - loss: 0.5434 - acc: 0.7242 - val_loss: 0.6721 - val_acc: 0.51
47
Epoch 12/30
22/22 [==============================] - 4s 188ms/step - loss: 0.5483 - acc: 0.7341 - val_loss: 0.6108 - val_acc: 0.79
41
Epoch 13/30
22/22 [==============================] - 4s 194ms/step - loss: 0.5318 - acc: 0.7310 - val_loss: 0.6412 - val_acc: 0.58
33
Epoch 14/30
22/22 [==============================] - 4s 187ms/step - loss: 0.5318 - acc: 0.7384 - val_loss: 0.5849 - val_acc: 0.70
59
Epoch 15/30
22/22 [==============================] - 4s 187ms/step - loss: 0.5099 - acc: 0.7653 - val_loss: 0.6333 - val_acc: 0.58
82
Epoch 16/30
22/22 [==============================] - 4s 200ms/step - loss: 0.4799 - acc: 0.7640 - val_loss: 0.5118 - val_acc: 0.79
41
Epoch 17/30
22/22 [==============================] - 5s 221ms/step - loss: 0.4865 - acc: 0.7523 - val_loss: 0.5687 - val_acc: 0.75
00
Epoch 18/30
22/22 [==============================] - 5s 208ms/step - loss: 0.4621 - acc: 0.7795 - val_loss: 0.4880 - val_acc: 0.82
35
Epoch 19/30
22/22 [==============================] - 4s 188ms/step - loss: 0.4681 - acc: 0.7667 - val_loss: 0.6004 - val_acc: 0.67
65
Epoch 20/30
22/22 [==============================] - 4s 188ms/step - loss: 0.4470 - acc: 0.7895 - val_loss: 0.5915 - val_acc: 0.69
12
Epoch 21/30
22/22 [==============================] - 4s 191ms/step - loss: 0.4219 - acc: 0.8177 - val_loss: 0.5652 - val_acc: 0.72
92
Epoch 22/30
22/22 [==============================] - 4s 187ms/step - loss: 0.4322 - acc: 0.8094 - val_loss: 0.6375 - val_acc: 0.63
24
Epoch 23/30
22/22 [==============================] - 4s 188ms/step - loss: 0.3735 - acc: 0.8520 - val_loss: 0.5186 - val_acc: 0.77
94
Epoch 24/30
22/22 [==============================] - 4s 192ms/step - loss: 0.3640 - acc: 0.8565 - val_loss: 0.5752 - val_acc: 0.72
06
Epoch 25/30
22/22 [==============================] - 5s 225ms/step - loss: 0.3552 - acc: 0.8420 - val_loss: 0.5459 - val_acc: 0.75
00
Epoch 26/30
22/22 [==============================] - 4s 188ms/step - loss: 0.3412 - acc: 0.8578 - val_loss: 0.6166 - val_acc: 0.64
71
Epoch 27/30
22/22 [==============================] - 4s 186ms/step - loss: 0.3232 - acc: 0.8593 - val_loss: 0.5564 - val_acc: 0.75
00
Epoch 28/30
22/22 [==============================] - 4s 187ms/step - loss: 0.3248 - acc: 0.8606 - val_loss: 0.5707 - val_acc: 0.76
47
Epoch 29/30
22/22 [==============================] - 4s 190ms/step - loss: 0.2972 - acc: 0.8805 - val_loss: 0.5680 - val_acc: 0.72
92
Epoch 30/30
22/22 [==============================] - 4s 186ms/step - loss: 0.2969 - acc: 0.8948 - val_loss: 0.5141 - val_acc: 0.72
06

testing
------------------------
```

```
4952/4952 [==============================] - 72s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  pottedplant
3   007032.jpg            1
4   002185.jpg            1
7   009739.jpg            1
9   005722.jpg            1
12  005927.jpg            1
Samples of rightly predicted as one
-----------------------
['009514.jpg', '000389.jpg', '005174.jpg', '003756.jpg', '006750.jpg']
actual ones:  254
predicted ones:  1731
Matched ones(True Positive):  145
prediction accuracy of ones:  57.08661417322835 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '008839.jpg', '001210.jpg']
actual zeros:  4698
predicted zeros:  3221
Matched zeros (True Negative):  3112
prediction accuracy of zeros:  66.24095359727544 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 3257
Overall prediction accuracy:  65.77140549273021 %
```

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=30 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_r = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_r = get_result_db(imageMap_test, test_generator, pred_r, columns, column)
```

```
training
-------------------------
Epoch 1/30
23/23 [==============================] - 7s 313ms/step - loss: 0.6990 - acc: 0.6399 - val_loss: 0.6738 - val_acc: 0.69
79
Epoch 2/30
23/23 [==============================] - 4s 169ms/step - loss: 0.6327 - acc: 0.6853 - val_loss: 0.6687 - val_acc: 0.70
59
Epoch 3/30
23/23 [==============================] - 5s 217ms/step - loss: 0.6216 - acc: 0.6744 - val_loss: 0.6513 - val_acc: 0.73
53
Epoch 4/30
23/23 [==============================] - 5s 209ms/step - loss: 0.6201 - acc: 0.6642 - val_loss: 0.6428 - val_acc: 0.69
12
Epoch 5/30
23/23 [==============================] - 4s 190ms/step - loss: 0.6030 - acc: 0.6873 - val_loss: 0.6332 - val_acc: 0.71
88
Epoch 6/30
23/23 [==============================] - 4s 186ms/step - loss: 0.5960 - acc: 0.6771 - val_loss: 0.5762 - val_acc: 0.73
53
Epoch 7/30
23/23 [==============================] - 4s 186ms/step - loss: 0.5802 - acc: 0.6907 - val_loss: 0.6342 - val_acc: 0.66
18
Epoch 8/30
23/23 [==============================] - 4s 187ms/step - loss: 0.5611 - acc: 0.6927 - val_loss: 0.5788 - val_acc: 0.75
00
Epoch 9/30
23/23 [==============================] - 4s 190ms/step - loss: 0.5690 - acc: 0.6914 - val_loss: 0.6241 - val_acc: 0.70
83
Epoch 10/30
23/23 [==============================] - 4s 184ms/step - loss: 0.5503 - acc: 0.6975 - val_loss: 0.6477 - val_acc: 0.70
59
Epoch 11/30
23/23 [==============================] - 4s 186ms/step - loss: 0.5676 - acc: 0.7036 - val_loss: 0.6360 - val_acc: 0.73
53
Epoch 12/30
23/23 [==============================] - 4s 183ms/step - loss: 0.4991 - acc: 0.7539 - val_loss: 0.6402 - val_acc: 0.66
18
Epoch 13/30
23/23 [==============================] - 4s 187ms/step - loss: 0.5133 - acc: 0.7532 - val_loss: 0.6154 - val_acc: 0.71
88
Epoch 14/30
23/23 [==============================] - 4s 185ms/step - loss: 0.4903 - acc: 0.7763 - val_loss: 0.6936 - val_acc: 0.52
94
Epoch 15/30
23/23 [==============================] - 4s 184ms/step - loss: 0.5014 - acc: 0.7661 - val_loss: 0.6602 - val_acc: 0.55
88
Epoch 16/30
23/23 [==============================] - 4s 184ms/step - loss: 0.4589 - acc: 0.7906 - val_loss: 0.6246 - val_acc: 0.64
71
Epoch 17/30
23/23 [==============================] - 4s 189ms/step - loss: 0.4560 - acc: 0.7886 - val_loss: 0.6536 - val_acc: 0.62
50
Epoch 18/30
23/23 [==============================] - 4s 186ms/step - loss: 0.4330 - acc: 0.7879 - val_loss: 0.6868 - val_acc: 0.55
88
Epoch 19/30
23/23 [==============================] - 4s 185ms/step - loss: 0.4267 - acc: 0.8090 - val_loss: 0.6531 - val_acc: 0.66
18
Epoch 20/30
23/23 [==============================] - 4s 185ms/step - loss: 0.4243 - acc: 0.8056 - val_loss: 0.6930 - val_acc: 0.61
76
Epoch 21/30
23/23 [==============================] - 5s 211ms/step - loss: 0.3653 - acc: 0.8572 - val_loss: 0.6513 - val_acc: 0.69
79
Epoch 22/30
23/23 [==============================] - 5s 214ms/step - loss: 0.3792 - acc: 0.8362 - val_loss: 0.7217 - val_acc: 0.55
88
Epoch 23/30
23/23 [==============================] - 4s 193ms/step - loss: 0.3835 - acc: 0.8253 - val_loss: 0.6239 - val_acc: 0.72
06
Epoch 24/30
23/23 [==============================] - 4s 184ms/step - loss: 0.3555 - acc: 0.8443 - val_loss: 0.6973 - val_acc: 0.63
24
Epoch 25/30
23/23 [==============================] - 5s 214ms/step - loss: 0.3289 - acc: 0.8573 - val_loss: 0.6771 - val_acc: 0.66
67
Epoch 26/30
23/23 [==============================] - 5s 199ms/step - loss: 0.3154 - acc: 0.8627 - val_loss: 0.6637 - val_acc: 0.64
71
Epoch 27/30
23/23 [==============================] - 4s 184ms/step - loss: 0.3027 - acc: 0.8681 - val_loss: 0.7624 - val_acc: 0.60
29
Epoch 28/30
23/23 [==============================] - 4s 184ms/step - loss: 0.2876 - acc: 0.9007 - val_loss: 0.7521 - val_acc: 0.73
53
Epoch 29/30
23/23 [==============================] - 4s 187ms/step - loss: 0.2625 - acc: 0.8967 - val_loss: 0.7140 - val_acc: 0.73
96
Epoch 30/30
23/23 [==============================] - 4s 184ms/step - loss: 0.2725 - acc: 0.8851 - val_loss: 0.7051 - val_acc: 0.67
65

testing
-------------------------
```

```
4952/4952 [==============================] - 72s 14ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  pottedplant
4   002185.jpg            1
6   001210.jpg            1
8   006003.jpg            1
10  005155.jpg            1
13  009856.jpg            1
Samples of rightly predicted as one
-----------------------
['000006.jpg', '009514.jpg', '000389.jpg', '005174.jpg', '005866.jpg']
actual ones:  254
predicted ones:  1075
Matched ones(True Positive):  99
prediction accuracy of ones:  38.976377952755904 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '007032.jpg', '008839.jpg']
actual zeros:  4698
predicted zeros:  3877
Matched zeros (True Negative):  3722
prediction accuracy of zeros:  79.22520221370796 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 3821
Overall prediction accuracy:  77.16074313408724 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n-------------------------')
classified_array = np.array(results_q[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
-------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplan
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0	(
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	(
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	(
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	(
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1	
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	(
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c15.csv')
```

Training Column 16

In [0]:
```
column = 16
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : sheep
Number of positive samples:  97
Number of selected negative samples:  211
Number of merged samples:  308
For Training:
Found 206 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	000416.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
1	005414.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
2	006330.jpg	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
3	002696.jpg	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	
4	000073.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	
5	002683.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
6	004010.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
7	000654.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	
8	006073.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	
9	007859.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_s = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_s = get_result_db(imageMap_test, test_generator, pred_s, columns, column)
```

```
training
-------------------------
Epoch 1/10
6/6 [==============================] - 6s 929ms/step - loss: 0.8408 - acc: 0.6146 - val_loss: 0.6601 - val_acc: 0.6771
Epoch 2/10
6/6 [==============================] - 0s 61ms/step - loss: 0.6457 - acc: 0.6590 - val_loss: 0.6200 - val_acc: 0.7206
Epoch 3/10
6/6 [==============================] - 0s 39ms/step - loss: 0.6476 - acc: 0.6590 - val_loss: 0.6642 - val_acc: 0.7059
Epoch 4/10
6/6 [==============================] - 1s 238ms/step - loss: 0.5634 - acc: 0.7397 - val_loss: 0.6211 - val_acc: 0.6765
Epoch 5/10
6/6 [==============================] - 1s 212ms/step - loss: 0.6468 - acc: 0.6427 - val_loss: 0.6388 - val_acc: 0.7292
Epoch 6/10
6/6 [==============================] - 1s 198ms/step - loss: 0.5969 - acc: 0.6749 - val_loss: 0.6289 - val_acc: 0.7647
Epoch 7/10
6/6 [==============================] - 1s 199ms/step - loss: 0.5605 - acc: 0.6854 - val_loss: 0.6409 - val_acc: 0.6912
Epoch 8/10
6/6 [==============================] - 1s 217ms/step - loss: 0.5527 - acc: 0.7188 - val_loss: 0.5657 - val_acc: 0.7794
Epoch 9/10
6/6 [==============================] - 1s 215ms/step - loss: 0.5094 - acc: 0.7440 - val_loss: 0.5565 - val_acc: 0.7604
Epoch 10/10
6/6 [==============================] - 1s 197ms/step - loss: 0.5577 - acc: 0.7119 - val_loss: 0.5507 - val_acc: 0.7941

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  sheep
0   001019.jpg      1
2   000371.jpg      1
6   001210.jpg      1
13  009856.jpg      1
25  009229.jpg      1
Samples of rightly predicted as one
----------------------
['000175.jpg', '004582.jpg', '005915.jpg', '000574.jpg', '009031.jpg']
actual ones:  98
predicted ones:  955
Matched ones(True Positive):  69
prediction accuracy of ones:  70.40816326530613 %


Samples of rightly predicted as zero
----------------------
['001639.jpg', '007032.jpg', '002185.jpg', '008839.jpg', '009739.jpg']
actual zeros:  4854
predicted zeros:  3997
Matched zeros (True Negative):  3968
prediction accuracy of zeros:  81.74701277297075 %


Rightly predicted overall
----------------------
Number of samples 4952
Overall Matches 4037
Overall prediction accuracy:  81.52261712439419 %
```

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=20 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_t = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_t = get_result_db(imageMap_test, test_generator, pred_t, columns, column)
```

```
training
-------------------------
Epoch 1/20
6/6 [==============================] - 6s 967ms/step - loss: 0.7836 - acc: 0.6450 - val_loss: 0.6665 - val_acc: 0.6324
Epoch 2/20
6/6 [==============================] - 0s 39ms/step - loss: 0.6422 - acc: 0.7247 - val_loss: 0.6114 - val_acc: 0.7353
Epoch 3/20
6/6 [==============================] - 0s 76ms/step - loss: 0.6474 - acc: 0.6863 - val_loss: 0.6449 - val_acc: 0.6765
Epoch 4/20
6/6 [==============================] - 1s 232ms/step - loss: 0.6424 - acc: 0.6802 - val_loss: 0.6496 - val_acc: 0.6771
Epoch 5/20
6/6 [==============================] - 1s 199ms/step - loss: 0.6282 - acc: 0.6811 - val_loss: 0.6435 - val_acc: 0.6618
Epoch 6/20
6/6 [==============================] - 1s 217ms/step - loss: 0.6292 - acc: 0.6875 - val_loss: 0.6380 - val_acc: 0.7059
Epoch 7/20
6/6 [==============================] - 1s 199ms/step - loss: 0.6137 - acc: 0.6965 - val_loss: 0.6568 - val_acc: 0.6765
Epoch 8/20
6/6 [==============================] - 1s 216ms/step - loss: 0.6173 - acc: 0.6599 - val_loss: 0.6450 - val_acc: 0.6875
Epoch 9/20
6/6 [==============================] - 1s 202ms/step - loss: 0.5865 - acc: 0.6978 - val_loss: 0.6375 - val_acc: 0.6765
Epoch 10/20
6/6 [==============================] - 1s 199ms/step - loss: 0.6028 - acc: 0.7089 - val_loss: 0.6182 - val_acc: 0.6912
Epoch 11/20
6/6 [==============================] - 1s 197ms/step - loss: 0.5393 - acc: 0.6868 - val_loss: 0.6265 - val_acc: 0.8088
Epoch 12/20
6/6 [==============================] - 1s 214ms/step - loss: 0.5569 - acc: 0.7022 - val_loss: 0.5617 - val_acc: 0.7188
Epoch 13/20
6/6 [==============================] - 1s 216ms/step - loss: 0.5375 - acc: 0.7031 - val_loss: 0.5642 - val_acc: 0.6471
Epoch 14/20
6/6 [==============================] - 1s 197ms/step - loss: 0.6192 - acc: 0.6788 - val_loss: 0.5742 - val_acc: 0.7794
Epoch 15/20
6/6 [==============================] - 1s 199ms/step - loss: 0.5061 - acc: 0.7070 - val_loss: 0.6362 - val_acc: 0.6324
Epoch 16/20
6/6 [==============================] - 1s 214ms/step - loss: 0.5179 - acc: 0.7387 - val_loss: 0.5461 - val_acc: 0.8229
Epoch 17/20
6/6 [==============================] - 1s 198ms/step - loss: 0.4973 - acc: 0.7079 - val_loss: 0.5297 - val_acc: 0.8382
Epoch 18/20
6/6 [==============================] - 1s 198ms/step - loss: 0.4671 - acc: 0.7776 - val_loss: 0.6096 - val_acc: 0.7059
Epoch 19/20
6/6 [==============================] - 1s 227ms/step - loss: 0.4783 - acc: 0.7714 - val_loss: 0.5607 - val_acc: 0.7500
Epoch 20/20
6/6 [==============================] - 2s 261ms/step - loss: 0.4763 - acc: 0.7656 - val_loss: 0.5703 - val_acc: 0.7188

testing
-------------------------
4952/4952 [==============================] - 74s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
    Filenames  sheep
0  001019.jpg      1
2  000371.jpg      1
5  008839.jpg      1
6  001210.jpg      1
7  009739.jpg      1
Samples of rightly predicted as one
-----------------------
['003152.jpg', '000175.jpg', '004582.jpg', '005915.jpg', '000574.jpg']
actual ones:  98
predicted ones:  2159
Matched ones(True Positive):  81
prediction accuracy of ones:  82.6530612244898 %

Samples of rightly predicted as zero
-----------------------
['001639.jpg', '007032.jpg', '002185.jpg', '006003.jpg', '005155.jpg']
actual zeros:  4854
predicted zeros:  2793
Matched zeros (True Negative):  2776
prediction accuracy of zeros:  57.18994643592913 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 2857
Overall prediction accuracy:  57.69386106623586 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
classified_array = np.array(results_s[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplan
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0	
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1	
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c16.csv')
```

Training Column 17

In [0]:
```
column = 17
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : sofa
Number of positive samples:  372
Number of selected negative samples:  819
Number of merged samples:  1191
For Training:
Found 1074 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	002559.jpg	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	
1	005429.jpg	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
2	003877.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	
3	000476.jpg	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	
4	008987.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
5	002300.jpg	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
6	005292.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	008315.jpg	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	
8	006424.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
9	006638.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_v = model.predict_generator(test_generator,
                                 steps=STEP_SIZE_TEST,
                                 #steps = 100,
                                 verbose=1)

print('\nResult showcasing\n-------------------------')
results_v = get_result_db(imageMap_test, test_generator, pred_v, columns, column)
```

```
training
-------------------------
Epoch 1/10
33/33 [==============================] - 10s 313ms/step - loss: 0.6804 - acc: 0.6407 - val_loss: 0.6096 - val_acc: 0.7
059
Epoch 2/10
33/33 [==============================] - 6s 172ms/step - loss: 0.6134 - acc: 0.6724 - val_loss: 0.5974 - val_acc: 0.70
59
Epoch 3/10
33/33 [==============================] - 6s 185ms/step - loss: 0.5355 - acc: 0.7302 - val_loss: 0.5730 - val_acc: 0.67
65
Epoch 4/10
33/33 [==============================] - 6s 189ms/step - loss: 0.5871 - acc: 0.6874 - val_loss: 0.6307 - val_acc: 0.66
67
Epoch 5/10
33/33 [==============================] - 6s 184ms/step - loss: 0.5464 - acc: 0.7023 - val_loss: 0.6979 - val_acc: 0.52
94
Epoch 6/10
33/33 [==============================] - 6s 183ms/step - loss: 0.5346 - acc: 0.6952 - val_loss: 0.5991 - val_acc: 0.64
71
Epoch 7/10
33/33 [==============================] - 6s 184ms/step - loss: 0.5133 - acc: 0.7338 - val_loss: 0.5262 - val_acc: 0.75
00
Epoch 8/10
33/33 [==============================] - 6s 185ms/step - loss: 0.5384 - acc: 0.7217 - val_loss: 0.5513 - val_acc: 0.66
67
Epoch 9/10
33/33 [==============================] - 6s 184ms/step - loss: 0.5093 - acc: 0.7463 - val_loss: 0.5616 - val_acc: 0.67
65
Epoch 10/10
33/33 [==============================] - 6s 184ms/step - loss: 0.5076 - acc: 0.7435 - val_loss: 0.5492 - val_acc: 0.72
06

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  sofa
3   007032.jpg     1
11  004599.jpg     1
14  008889.jpg     1
17  004226.jpg     1
23  009075.jpg     1
Samples of rightly predicted as one
-----------------------
['007032.jpg', '006487.jpg', '008486.jpg', '005801.jpg', '003096.jpg']
actual ones:  355
predicted ones:  759
Matched ones(True Positive):  141
prediction accuracy of ones:  39.718309859154935 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '002185.jpg', '008839.jpg']
actual zeros:  4597
predicted zeros:  4193
Matched zeros (True Negative):  3979
prediction accuracy of zeros:  86.55644985860343 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 4120
Overall prediction accuracy:  83.19870759289176 %
```

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_u = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_u = get_result_db(imageMap_test, test_generator, pred_u, columns, column)
```

```
training
-------------------------
Epoch 1/10
33/33 [==============================] - 10s 289ms/step - loss: 0.6182 - acc: 0.6686 - val_loss: 0.6299 - val_acc: 0.6
771
Epoch 2/10
33/33 [==============================] - 6s 170ms/step - loss: 0.5896 - acc: 0.6935 - val_loss: 0.5360 - val_acc: 0.73
53
Epoch 3/10
33/33 [==============================] - 6s 186ms/step - loss: 0.5714 - acc: 0.6826 - val_loss: 0.5674 - val_acc: 0.70
59
Epoch 4/10
33/33 [==============================] - 7s 206ms/step - loss: 0.5434 - acc: 0.7032 - val_loss: 0.5810 - val_acc: 0.69
12
Epoch 5/10
33/33 [==============================] - 7s 222ms/step - loss: 0.5558 - acc: 0.6909 - val_loss: 0.5472 - val_acc: 0.70
83
Epoch 6/10
33/33 [==============================] - 7s 198ms/step - loss: 0.5400 - acc: 0.7224 - val_loss: 0.6106 - val_acc: 0.67
65
Epoch 7/10
33/33 [==============================] - 6s 184ms/step - loss: 0.5266 - acc: 0.7212 - val_loss: 0.5731 - val_acc: 0.73
53
Epoch 8/10
33/33 [==============================] - 6s 184ms/step - loss: 0.5142 - acc: 0.7430 - val_loss: 0.6595 - val_acc: 0.64
71
Epoch 9/10
33/33 [==============================] - 6s 185ms/step - loss: 0.5096 - acc: 0.7354 - val_loss: 0.5745 - val_acc: 0.71
88
Epoch 10/10
33/33 [==============================] - 6s 185ms/step - loss: 0.4796 - acc: 0.7539 - val_loss: 0.5805 - val_acc: 0.72
06

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  sofa
3   007032.jpg     1
7   009739.jpg     1
9   005722.jpg     1
11  004599.jpg     1
14  008889.jpg     1
Samples of rightly predicted as one
-----------------------
['007032.jpg', '009739.jpg', '003224.jpg', '006487.jpg', '008486.jpg']
actual ones:  355
predicted ones:  1338
Matched ones(True Positive):  226
prediction accuracy of ones:  63.66197183098592 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '002185.jpg', '008839.jpg']
actual zeros:  4597
predicted zeros:  3614
Matched zeros (True Negative):  3485
prediction accuracy of zeros:  75.81031107243855 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 3711
Overall prediction accuracy:  74.93941841680129 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
classified_array = np.array(results_u[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplan
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0	(
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	(
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	(
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	(
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1	
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	(
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c17.csv')
```

Training Column 18

In [0]:
```
column = 18
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : train
Number of positive samples:  263
Number of selected negative samples:  572
Number of merged samples:  835
For Training:
Found 726 images.
For Validation:
Found 100 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	007421.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	004283.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
2	009790.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	006627.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	
4	003034.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	
5	009494.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
6	002525.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	002767.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	
8	002738.jpg	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	003939.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=15 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_x = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_x = get_result_db(imageMap_test, test_generator, pred_x, columns, column)
```

```
training
------------------------
Epoch 1/15
22/22 [==============================] - 8s 366ms/step - loss: 0.6628 - acc: 0.6413 - val_loss: 0.6365 - val_acc: 0.69
12
Epoch 2/15
22/22 [==============================] - 4s 162ms/step - loss: 0.5948 - acc: 0.7033 - val_loss: 0.5984 - val_acc: 0.67
65
Epoch 3/15
22/22 [==============================] - 4s 193ms/step - loss: 0.5853 - acc: 0.7027 - val_loss: 0.5675 - val_acc: 0.75
00
Epoch 4/15
22/22 [==============================] - 5s 219ms/step - loss: 0.5548 - acc: 0.7106 - val_loss: 0.5877 - val_acc: 0.75
00
Epoch 5/15
22/22 [==============================] - 5s 209ms/step - loss: 0.5350 - acc: 0.7476 - val_loss: 0.5491 - val_acc: 0.77
94
Epoch 6/15
22/22 [==============================] - 4s 184ms/step - loss: 0.5216 - acc: 0.7508 - val_loss: 0.5966 - val_acc: 0.67
65
Epoch 7/15
22/22 [==============================] - 4s 184ms/step - loss: 0.4906 - acc: 0.7701 - val_loss: 0.5276 - val_acc: 0.75
00
Epoch 8/15
22/22 [==============================] - 4s 188ms/step - loss: 0.5041 - acc: 0.7638 - val_loss: 0.6359 - val_acc: 0.65
62
Epoch 9/15
22/22 [==============================] - 4s 185ms/step - loss: 0.4872 - acc: 0.7794 - val_loss: 0.6306 - val_acc: 0.61
76
Epoch 10/15
22/22 [==============================] - 4s 185ms/step - loss: 0.4638 - acc: 0.7857 - val_loss: 0.5393 - val_acc: 0.73
53
Epoch 11/15
22/22 [==============================] - 4s 184ms/step - loss: 0.4735 - acc: 0.7875 - val_loss: 0.5823 - val_acc: 0.73
53
Epoch 12/15
22/22 [==============================] - 4s 188ms/step - loss: 0.4478 - acc: 0.8000 - val_loss: 0.5860 - val_acc: 0.68
75
Epoch 13/15
22/22 [==============================] - 4s 183ms/step - loss: 0.4302 - acc: 0.8020 - val_loss: 0.5496 - val_acc: 0.72
06
Epoch 14/15
22/22 [==============================] - 4s 186ms/step - loss: 0.4189 - acc: 0.7938 - val_loss: 0.6894 - val_acc: 0.57
35
Epoch 15/15
22/22 [==============================] - 4s 186ms/step - loss: 0.4154 - acc: 0.8139 - val_loss: 0.6743 - val_acc: 0.60
29

testing
------------------------
4952/4952 [==============================] - 74s 15ms/step

Result showcasing
------------------------
Predicted Results (for ones only)--------------
     Filenames  train
0   001019.jpg      1
2   000371.jpg      1
5   008839.jpg      1
6   001210.jpg      1
10  005155.jpg      1
Samples of rightly predicted as one
----------------------
['007037.jpg', '003286.jpg', '002148.jpg', '006763.jpg', '000629.jpg']
actual ones:  259
predicted ones:  2471
Matched ones(True Positive):  234
prediction accuracy of ones:  90.34749034749035 %


Samples of rightly predicted as zero
----------------------
['001639.jpg', '007032.jpg', '002185.jpg', '009739.jpg', '006003.jpg']
actual zeros:  4693
predicted zeros:  2481
Matched zeros (True Negative):  2456
prediction accuracy of zeros:  52.333262305561476 %


Rightly predicted overall
----------------------
Number of samples 4952
Overall Matches 2690
Overall prediction accuracy:  54.32148626817448 %
```

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_w = model.predict_generator(test_generator,
                             steps=STEP_SIZE_TEST,
                             #steps = 100,
                             verbose=1)

print('\nResult showcasing\n-------------------------')
results_w = get_result_db(imageMap_test, test_generator, pred_w, columns, column)
```

```
training
-------------------------
Epoch 1/10
22/22 [==============================] - 8s 366ms/step - loss: 0.6839 - acc: 0.6548 - val_loss: 0.6071 - val_acc: 0.69
79
Epoch 2/10
22/22 [==============================] - 4s 166ms/step - loss: 0.6144 - acc: 0.6710 - val_loss: 0.6105 - val_acc: 0.67
65
Epoch 3/10
22/22 [==============================] - 4s 186ms/step - loss: 0.5626 - acc: 0.7419 - val_loss: 0.5350 - val_acc: 0.69
12
Epoch 4/10
22/22 [==============================] - 4s 186ms/step - loss: 0.5292 - acc: 0.7403 - val_loss: 0.5728 - val_acc: 0.69
12
Epoch 5/10
22/22 [==============================] - 4s 190ms/step - loss: 0.5075 - acc: 0.7468 - val_loss: 0.5074 - val_acc: 0.76
04
Epoch 6/10
22/22 [==============================] - 4s 185ms/step - loss: 0.5000 - acc: 0.7894 - val_loss: 0.5622 - val_acc: 0.67
65
Epoch 7/10
22/22 [==============================] - 4s 186ms/step - loss: 0.4857 - acc: 0.7815 - val_loss: 0.4915 - val_acc: 0.77
94
Epoch 8/10
22/22 [==============================] - 4s 186ms/step - loss: 0.4782 - acc: 0.7691 - val_loss: 0.5529 - val_acc: 0.72
06
Epoch 9/10
22/22 [==============================] - 5s 222ms/step - loss: 0.4573 - acc: 0.7946 - val_loss: 0.5552 - val_acc: 0.69
79
Epoch 10/10
22/22 [==============================] - 5s 224ms/step - loss: 0.4305 - acc: 0.8089 - val_loss: 0.4502 - val_acc: 0.79
41

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  train
6   001210.jpg      1
18  007037.jpg      1
19  003286.jpg      1
30  000004.jpg      1
31  002148.jpg      1
Samples of rightly predicted as one
-----------------------
['007037.jpg', '003286.jpg', '002148.jpg', '006763.jpg', '006356.jpg']
actual ones:  259
predicted ones:  639
Matched ones(True Positive):  154
prediction accuracy of ones:  59.45945945945946 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '001639.jpg', '000371.jpg', '007032.jpg', '002185.jpg']
actual zeros:  4693
predicted zeros:  4313
Matched zeros (True Negative):  4208
prediction accuracy of zeros:  89.66545919454506 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 4362
Overall prediction accuracy:  88.08562197092084 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
classified_array = np.array(results_w[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplan
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0	0
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1	
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	

In [0]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c18.csv')
```

Training Column 19

In [0]:
```
column = 19
modifiedDb2, train_generator, valid_generator, test_generator = generate_modified_db(imageMap_trainval, imageMap_test,
columns, column)
modifiedDb2[:10]
```

```
Avoided : tvmonitor
Number of positive samples:  279
Number of selected negative samples:  610
Number of merged samples:  889
For Training:
Found 777 images.
For Validation:
Found 99 images.
For Testing:
Found 4952 images.
```

Out[0]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	003654.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
1	005605.jpg	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	
2	007299.jpg	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	004321.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
4	002124.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
5	009249.jpg	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
6	000843.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
7	006588.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
8	000967.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
9	000702.jpg	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	

In [0]:
```
model = build_model()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_y = model.predict_generator(test_generator,
                            steps=STEP_SIZE_TEST,
                            #steps = 100,
                            verbose=1)

print('\nResult showcasing\n-------------------------')
results_y = get_result_db(imageMap_test, test_generator, pred_y, columns, column)
```

```
training
-------------------------
Epoch 1/10
24/24 [==============================] - 9s 363ms/step - loss: 0.6477 - acc: 0.6719 - val_loss: 0.6517 - val_acc: 0.61
46
Epoch 2/10
24/24 [==============================] - 4s 185ms/step - loss: 0.6154 - acc: 0.6899 - val_loss: 0.6462 - val_acc: 0.61
19
Epoch 3/10
24/24 [==============================] - 4s 184ms/step - loss: 0.5894 - acc: 0.7042 - val_loss: 0.6422 - val_acc: 0.61
19
Epoch 4/10
24/24 [==============================] - 4s 184ms/step - loss: 0.5999 - acc: 0.6820 - val_loss: 0.6361 - val_acc: 0.62
69
Epoch 5/10
24/24 [==============================] - 5s 188ms/step - loss: 0.5692 - acc: 0.7049 - val_loss: 0.6352 - val_acc: 0.62
50
Epoch 6/10
24/24 [==============================] - 4s 185ms/step - loss: 0.5748 - acc: 0.7081 - val_loss: 0.6130 - val_acc: 0.67
16
Epoch 7/10
24/24 [==============================] - 4s 181ms/step - loss: 0.5517 - acc: 0.7271 - val_loss: 0.6133 - val_acc: 0.59
70
Epoch 8/10
24/24 [==============================] - 4s 182ms/step - loss: 0.5367 - acc: 0.7368 - val_loss: 0.6458 - val_acc: 0.62
69
Epoch 9/10
24/24 [==============================] - 4s 187ms/step - loss: 0.4963 - acc: 0.7649 - val_loss: 0.6067 - val_acc: 0.62
50
Epoch 10/10
24/24 [==============================] - 4s 182ms/step - loss: 0.5171 - acc: 0.7694 - val_loss: 0.5695 - val_acc: 0.74
63

testing
-------------------------
4952/4952 [==============================] - 73s 15ms/step

Result showcasing
-------------------------
Predicted Results (for ones only)---------------
     Filenames  tvmonitor
1   001639.jpg          1
5   008839.jpg          1
7   009739.jpg          1
15  007835.jpg          1
21  009492.jpg          1
Samples of rightly predicted as one
-----------------------
['007835.jpg', '008486.jpg', '005935.jpg', '008407.jpg', '000659.jpg']
actual ones:  255
predicted ones:  1414
Matched ones(True Positive):  149
prediction accuracy of ones:  58.43137254901961 %


Samples of rightly predicted as zero
-----------------------
['001019.jpg', '000371.jpg', '007032.jpg', '002185.jpg', '001210.jpg']
actual zeros:  4697
predicted zeros:  3538
Matched zeros (True Negative):  3432
prediction accuracy of zeros:  73.06791569086651 %


Rightly predicted overall
-----------------------
Number of samples 4952
Overall Matches 3581
Overall prediction accuracy:  72.31421647819063 %
```

In [0]:
```
print('\nUpdated Prediction DataFrame with new column\n------------------------')
classified_array = np.array(results_y[columns[column]])
results2[columns[column]] = classified_array
results2[:10]
```

```
Updated Prediction DataFrame with new column
------------------------
```

Out[0]:

	Unnamed: 0	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplan
0	0	001966.jpg	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0	
1	1	003029.jpg	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
2	2	001600.jpg	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	
3	3	000994.jpg	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	
4	4	000097.jpg	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	
5	5	006716.jpg	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
6	6	003378.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	7	001744.jpg	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1	
8	8	000231.jpg	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	
9	9	002703.jpg	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	

Result & Discussion

In [220]:
```
results2.to_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c19.csv')
result = pd.read_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c19.csv')
label = imageMap_test

prediction = result
actual = label
del prediction['Unnamed: 0.1']
del prediction['Unnamed: 0']

prediction = prediction.sort_values(by=['Filenames']).reset_index(drop=True)
actual = actual.sort_values(by=['Filenames']).reset_index(drop=True)

files = list(actual['Filenames'])
del prediction['Filenames']
del actual['Filenames']
pred_array = prediction.values
actual_array = actual.values
print('Total number of test samples: ',len(actual_array))

space = range(5,21)
for s in space:
    frac = int((s/20)*100)
    number = 0
    for n in range(len(actual_array)):
      if(((actual_array[n] == pred_array[n]).sum())>=s): number = number+1
    percent = number/len(actual_array)*100
    print('atleast ', frac,'% acurate predictions(',s,' out of 20) :',number,'  percentage: ',percent,'%')
```

```
Total number of test samples:  4952
atleast  25 % acurate predictions( 5  out of 20) : 4952   percentage:  100.0 %
atleast  30 % acurate predictions( 6  out of 20) : 4952   percentage:  100.0 %
atleast  35 % acurate predictions( 7  out of 20) : 4952   percentage:  100.0 %
atleast  40 % acurate predictions( 8  out of 20) : 4952   percentage:  100.0 %
atleast  45 % acurate predictions( 9  out of 20) : 4950   percentage:  99.95961227786752 %
atleast  50 % acurate predictions( 10  out of 20) : 4934   percentage:  99.63651050080774 %
atleast  55 % acurate predictions( 11  out of 20) : 4875   percentage:  98.44507269789983 %
atleast  60 % acurate predictions( 12  out of 20) : 4716   percentage:  95.23424878836833 %
atleast  65 % acurate predictions( 13  out of 20) : 4320   percentage:  87.23747980613894 %
atleast  70 % acurate predictions( 14  out of 20) : 3688   percentage:  74.47495961227787 %
atleast  75 % acurate predictions( 15  out of 20) : 2800   percentage:  56.54281098546042 %
atleast  80 % acurate predictions( 16  out of 20) : 1819   percentage:  36.73263327948303 %
atleast  85 % acurate predictions( 17  out of 20) : 921   percentage:  18.598546042003232 %
atleast  90 % acurate predictions( 18  out of 20) : 340   percentage:  6.8659127625201934 %
atleast  95 % acurate predictions( 19  out of 20) : 80   percentage:  1.615508885298869 %
atleast  100 % acurate predictions( 20  out of 20) : 10   percentage:  0.20193861066235863 %
```

In [221]:
```
for n in range(len(actual_array)):
      if(((actual_array[n] == pred_array[n]).sum())>19): print(files[n])
```

```
000300.jpg
000769.jpg
001368.jpg
002550.jpg
004893.jpg
006263.jpg
007235.jpg
007288.jpg
008458.jpg
009775.jpg
```

In [254]:
```
result = pd.read_csv('/content/drive/My Drive/Assignment6/Object Detection/results2c19.csv')
prediction = result
del prediction['Unnamed: 0.1']
del prediction['Unnamed: 0']
prediction = prediction.sort_values(by=['Filenames']).reset_index(drop=True)

#Lets see some examples of correct predictions
print(prediction[prediction['Filenames']=='007235.jpg'].to_string())
Image("/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images/007235.jpg")
```

```
      Filenames  aeroplane  bicycle  bird  boat  bottle  bus  car  cat  chair  cow  diningtable  dog  horse  motorbik
e  person  pottedplant  sheep  sofa  train  tvmonitor
3577  007235.jpg          0        1     0     0       0    0    0    0      0    0            0    0      0
0       1           0      0     0      0          0
```

Out[254]:

In [263]:
```
print(prediction[prediction['Filenames']=='008458.jpg'].to_string())
Image("/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images/008458.jpg")
```

```
      Filenames  aeroplane  bicycle  bird  boat  bottle  bus  car  cat  chair  cow  diningtable  dog  horse  motorbik
e  person  pottedplant  sheep  sofa  train  tvmonitor
4203  008458.jpg          0        0     0     0       1    0    0    0      0    0            0    0      0
0       1           1      0     0      0          0
```

Out[263]:

Model 02

Version 1

In [213]:
```
train_generator, valid_generator, test_generator = generate_modified_db2(imageMap_trainval, imageMap_test, columns)
```

```
For Training:
Found 2501 images.
For Validation:
Found 2510 images.
For Testing:
Found 4952 images.
```

In [214]:
```
model = build_model2()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_two = model.predict_generator(test_generator,
                                   steps=STEP_SIZE_TEST,
                                   #steps = 100,
                                   verbose=1)
```

```
training
-------------------------
Epoch 1/10
78/78 [==============================] - 28s 360ms/step - loss: 0.2858 - acc: 0.9112 - val_loss: 0.2958 - val_acc: 0.9
224
Epoch 2/10
78/78 [==============================] - 26s 339ms/step - loss: 0.2615 - acc: 0.9185 - val_loss: 0.2850 - val_acc: 0.9
223
Epoch 3/10
78/78 [==============================] - 25s 325ms/step - loss: 0.2507 - acc: 0.9191 - val_loss: 0.2521 - val_acc: 0.9
234
Epoch 4/10
78/78 [==============================] - 26s 335ms/step - loss: 0.2435 - acc: 0.9204 - val_loss: 0.2534 - val_acc: 0.9
239
Epoch 5/10
78/78 [==============================] - 26s 334ms/step - loss: 0.2369 - acc: 0.9205 - val_loss: 0.2382 - val_acc: 0.9
240
Epoch 6/10
78/78 [==============================] - 26s 330ms/step - loss: 0.2329 - acc: 0.9207 - val_loss: 0.2348 - val_acc: 0.9
239
Epoch 7/10
78/78 [==============================] - 27s 349ms/step - loss: 0.2278 - acc: 0.9225 - val_loss: 0.2314 - val_acc: 0.9
248
Epoch 8/10
78/78 [==============================] - 26s 332ms/step - loss: 0.2239 - acc: 0.9202 - val_loss: 0.2373 - val_acc: 0.9
244
Epoch 9/10
78/78 [==============================] - 26s 334ms/step - loss: 0.2179 - acc: 0.9231 - val_loss: 0.2292 - val_acc: 0.9
231
Epoch 10/10
78/78 [==============================] - 25s 323ms/step - loss: 0.2128 - acc: 0.9237 - val_loss: 0.2257 - val_acc: 0.9
224

testing
-------------------------
4952/4952 [==============================] - 71s 14ms/step
```

In [215]:
```
#Lets see how one prediction looks like
pred_two[0]
```

Out[215]:
```
array([0.24217302, 0.02163741, 0.02550492, 0.09204993, 0.03996256,
       0.10946915, 0.3679792 , 0.00872102, 0.05048174, 0.07002816,
       0.02134874, 0.02699575, 0.05776963, 0.05683506, 0.37834436,
       0.03133172, 0.03837329, 0.03227076, 0.28227705, 0.06678405],
      dtype=float32)
```

In [264]:
```
pred_bool = (pred_two >0.5)

predictions = pred_bool.astype(int)
#columns should be the same order of y_col
results = pd.DataFrame(predictions, columns=columns)
results["Filenames"] = test_generator.filenames
ordered_cols = ["Filenames"]+columns
results = results[ordered_cols]#To get the same column order
#results[results.bird==1]
results[:10]
```

Out[264]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	007157.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	002736.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
2	003624.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
3	002106.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	002927.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
5	008964.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	004589.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
7	001698.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	008567.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
9	005119.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

In [265]:
```
#Saveing to harddrive
results.to_csv('/content/drive/My Drive/Assignment6/Object Detection/result_model2.csv')

result = results
label = imageMap_test

prediction = result
actual = label

prediction = prediction.sort_values(by=['Filenames']).reset_index(drop=True)
actual = actual.sort_values(by=['Filenames']).reset_index(drop=True)

files = list(actual['Filenames'])
del prediction['Filenames']
del actual['Filenames']
pred_array = prediction.values
actual_array = actual.values
print('Total number of test samples: ',len(actual_array))

space = range(5,21)
for s in space:
    frac = int((s/20)*100)
    number = 0
    for n in range(len(actual_array)):
      if(((actual_array[n] == pred_array[n]).sum())>=s): number = number+1
    percent = number/len(actual_array)*100
    print('atleast ', frac,'% acurate predictions(',s,' out of 20) :',number,'  percentage: ',percent,'%')
```

```
Total number of test samples:  4952
atleast  25 % acurate predictions( 5  out of 20) : 4952   percentage:  100.0 %
atleast  30 % acurate predictions( 6  out of 20) : 4952   percentage:  100.0 %
atleast  35 % acurate predictions( 7  out of 20) : 4952   percentage:  100.0 %
atleast  40 % acurate predictions( 8  out of 20) : 4952   percentage:  100.0 %
atleast  45 % acurate predictions( 9  out of 20) : 4952   percentage:  100.0 %
atleast  50 % acurate predictions( 10  out of 20) : 4952   percentage:  100.0 %
atleast  55 % acurate predictions( 11  out of 20) : 4952   percentage:  100.0 %
atleast  60 % acurate predictions( 12  out of 20) : 4952   percentage:  100.0 %
atleast  65 % acurate predictions( 13  out of 20) : 4952   percentage:  100.0 %
atleast  70 % acurate predictions( 14  out of 20) : 4952   percentage:  100.0 %
atleast  75 % acurate predictions( 15  out of 20) : 4952   percentage:  100.0 %
atleast  80 % acurate predictions( 16  out of 20) : 4942   percentage:  99.79806138933765 %
atleast  85 % acurate predictions( 17  out of 20) : 4865   percentage:  98.24313408723748 %
atleast  90 % acurate predictions( 18  out of 20) : 4423   percentage:  89.31744749596123 %
atleast  95 % acurate predictions( 19  out of 20) : 2563   percentage:  51.75686591276252 %
atleast  100 % acurate predictions( 20  out of 20) : 445   percentage:  8.98626817447496 %
```

In [276]:
```
correct_pred = []
for n in range(len(actual_array)):
      if(((actual_array[n] == pred_array[n]).sum())>19): correct_pred.append(files[n])
correct_pred[430:440]
```

Out[276]:
```
['009505.jpg',
 '009553.jpg',
 '009590.jpg',
 '009610.jpg',
 '009612.jpg',
 '009622.jpg',
 '009648.jpg',
 '009653.jpg',
 '009714.jpg',
 '009736.jpg']
```

In [279]:
```
#Lets see some examples of correct predictions
print(result[result['Filenames']=='009505.jpg'].to_string())
Image("/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images/009505.jpg")
```

```
      Filenames  aeroplane  bicycle  bird  boat  bottle  bus  car  cat  chair  cow  diningtable  dog  horse  motorbik
e  person  pottedplant  sheep  sofa  train  tvmonitor
3510  009505.jpg          0        0     0     0       0    0    0    0      0    0            0    0      1
0       1            0      0     0      0          0
```

Out[279]:

Version 2

In [289]:
```
train_generator, valid_generator, test_generator = generate_modified_db2(imageMap_trainval, imageMap_test, columns)
```

```
For Training:
Found 4811 images.
For Validation:
Found 200 images.
For Testing:
Found 4952 images.
```

In [290]:
```
model = build_model2()
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

print('\ntraining\n-------------------------')
model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=10 #10
                   )
print('\ntesting\n-------------------------')
test_generator.reset()
pred_two = model.predict_generator(test_generator,
                                   steps=STEP_SIZE_TEST,
                                   #steps = 100,
                                   verbose=1)
```

```
training
-------------------------
Epoch 1/10
150/150 [==============================] - 31s 208ms/step - loss: 0.2790 - acc: 0.9147 - val_loss: 0.3134 - val_acc:
0.9227
Epoch 2/10
150/150 [==============================] - 28s 185ms/step - loss: 0.2542 - acc: 0.9198 - val_loss: 0.3078 - val_acc:
0.9205
Epoch 3/10
150/150 [==============================] - 28s 187ms/step - loss: 0.2423 - acc: 0.9208 - val_loss: 0.2769 - val_acc:
0.9205
Epoch 4/10
150/150 [==============================] - 29s 196ms/step - loss: 0.2364 - acc: 0.9211 - val_loss: 0.2530 - val_acc:
0.9244
Epoch 5/10
150/150 [==============================] - 30s 198ms/step - loss: 0.2289 - acc: 0.9227 - val_loss: 0.2480 - val_acc:
0.9205
Epoch 6/10
150/150 [==============================] - 29s 190ms/step - loss: 0.2249 - acc: 0.9229 - val_loss: 0.2396 - val_acc:
0.9235
Epoch 7/10
150/150 [==============================] - 29s 193ms/step - loss: 0.2204 - acc: 0.9231 - val_loss: 0.2211 - val_acc:
0.9253
Epoch 8/10
150/150 [==============================] - 28s 187ms/step - loss: 0.2140 - acc: 0.9248 - val_loss: 0.2298 - val_acc:
0.9260
Epoch 9/10
150/150 [==============================] - 29s 194ms/step - loss: 0.2090 - acc: 0.9262 - val_loss: 0.2203 - val_acc:
0.9259
Epoch 10/10
150/150 [==============================] - 29s 194ms/step - loss: 0.2057 - acc: 0.9265 - val_loss: 0.2156 - val_acc:
0.9241

testing
-------------------------
4952/4952 [==============================] - 71s 14ms/step
```

In [291]:
```
pred_bool = (pred_two >0.5)

predictions = pred_bool.astype(int)
#columns should be the same order of y_col
results = pd.DataFrame(predictions, columns=columns)
results["Filenames"] = test_generator.filenames
ordered_cols = ["Filenames"]+columns
results = results[ordered_cols]#To get the same column order
#results[results.bird==1]
results[:10]
```

Out[291]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	007157.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	002736.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	003624.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	002106.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	002927.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
5	008964.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	004589.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
7	001698.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	008567.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	005119.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [292]:
```
#Saveing to harddrive
result = results
label = imageMap_test

prediction = result
actual = label

prediction = prediction.sort_values(by=['Filenames']).reset_index(drop=True)
actual = actual.sort_values(by=['Filenames']).reset_index(drop=True)

files = list(actual['Filenames'])
del prediction['Filenames']
del actual['Filenames']
pred_array = prediction.values
actual_array = actual.values
print('Total number of test samples: ',len(actual_array))

space = range(5,21)
for s in space:
    frac = int((s/20)*100)
    number = 0
    for n in range(len(actual_array)):
      if(((actual_array[n] == pred_array[n]).sum())>=s): number = number+1
    percent = number/len(actual_array)*100
    print('atleast ', frac,'% acurate predictions(',s,' out of 20) :',number,'  percentage: ',percent,'%')
```

```
Total number of test samples:  4952
atleast  25 % acurate predictions( 5  out of 20) : 4952   percentage:  100.0 %
atleast  30 % acurate predictions( 6  out of 20) : 4952   percentage:  100.0 %
atleast  35 % acurate predictions( 7  out of 20) : 4952   percentage:  100.0 %
atleast  40 % acurate predictions( 8  out of 20) : 4952   percentage:  100.0 %
atleast  45 % acurate predictions( 9  out of 20) : 4952   percentage:  100.0 %
atleast  50 % acurate predictions( 10  out of 20) : 4952   percentage:  100.0 %
atleast  55 % acurate predictions( 11  out of 20) : 4952   percentage:  100.0 %
atleast  60 % acurate predictions( 12  out of 20) : 4952   percentage:  100.0 %
atleast  65 % acurate predictions( 13  out of 20) : 4952   percentage:  100.0 %
atleast  70 % acurate predictions( 14  out of 20) : 4952   percentage:  100.0 %
atleast  75 % acurate predictions( 15  out of 20) : 4952   percentage:  100.0 %
atleast  80 % acurate predictions( 16  out of 20) : 4945   percentage:  99.85864297253634 %
atleast  85 % acurate predictions( 17  out of 20) : 4864   percentage:  98.22294022617125 %
atleast  90 % acurate predictions( 18  out of 20) : 4427   percentage:  89.39822294022616 %
atleast  95 % acurate predictions( 19  out of 20) : 2818   percentage:  56.90630048465266 %
atleast  100 % acurate predictions( 20  out of 20) : 514   percentage:  10.379644588045235 %
```

Model 03

Version 1

In [361]:
```
train_generator, valid_generator, test_generator = generate_modified_db2(imageMap_trainval, imageMap_test, columns)
```

```
For Training:
Found 2501 images.
For Validation:
Found 2510 images.
For Testing:
Found 4952 images.
```

In [0]:
```
inp = Input(shape = (100,100,3))
x = Conv2D(32, (3, 3), padding = 'same')(inp)
x = Activation('relu')(x)
x = Conv2D(32, (3, 3))(x)
x = Activation('relu')(x)
x = MaxPooling2D(pool_size = (2, 2))(x)
x = Dropout(0.25)(x)
x = Conv2D(64, (3, 3), padding = 'same')(x)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3))(x)
x = Activation('relu')(x)
x = MaxPooling2D(pool_size = (2, 2))(x)
x = Dropout(0.25)(x)
x = Flatten()(x)
x = Dense(512)(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
output1 = Dense(1, activation = 'sigmoid')(x)
output2 = Dense(1, activation = 'sigmoid')(x)
output3 = Dense(1, activation = 'sigmoid')(x)
output4 = Dense(1, activation = 'sigmoid')(x)
output5 = Dense(1, activation = 'sigmoid')(x)
output6 = Dense(1, activation = 'sigmoid')(x)
output7 = Dense(1, activation = 'sigmoid')(x)
output8 = Dense(1, activation = 'sigmoid')(x)
output9 = Dense(1, activation = 'sigmoid')(x)
output10 = Dense(1, activation = 'sigmoid')(x)
output11 = Dense(1, activation = 'sigmoid')(x)
output12 = Dense(1, activation = 'sigmoid')(x)
output13 = Dense(1, activation = 'sigmoid')(x)
output14 = Dense(1, activation = 'sigmoid')(x)
output15 = Dense(1, activation = 'sigmoid')(x)
output16 = Dense(1, activation = 'sigmoid')(x)
output17 = Dense(1, activation = 'sigmoid')(x)
output18 = Dense(1, activation = 'sigmoid')(x)
output19 = Dense(1, activation = 'sigmoid')(x)
output20 = Dense(1, activation = 'sigmoid')(x)
model = Model(inp,[output1,output2,output3,output4,output5,output6,output7,
                   output8,output9,output10,output11,output12,output13,output14,
                   output15,output16,output17,output18,output19,output20])
model.compile(optimizers.rmsprop(lr = 0.0001, decay = 1e-6),
loss = ["binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy"],metrics = ["accuracy"])
```

In [0]:
```
def generator_wrapper(generator):
    for batch_x,batch_y in generator:
        yield (batch_x,[batch_y[:,i] for i in range(20)])
```

In [364]:
```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
model.fit_generator(generator = generator_wrapper(train_generator),
                    steps_per_epoch = STEP_SIZE_TRAIN,
                    validation_data = generator_wrapper(valid_generator),
                    validation_steps = STEP_SIZE_VALID,
                    epochs=10,verbose=1
                   )
```

```
Epoch 1/10
78/78 [==============================] - 33s 422ms/step - loss: 5.7988 - dense_63_loss: 0.2338 - dense_64_loss: 0.2438
- dense_65_loss: 0.2970 - dense_66_loss: 0.1936 - dense_67_loss: 0.2569 - dense_68_loss: 0.2046 - dense_69_loss: 0.499
3 - dense_70_loss: 0.2689 - dense_71_loss: 0.3856 - dense_72_loss: 0.1589 - dense_73_loss: 0.2364 - dense_74_loss: 0.3
248 - dense_75_loss: 0.2751 - dense_76_loss: 0.2412 - dense_77_loss: 0.7629 - dense_78_loss: 0.2911 - dense_79_loss:
0.1317 - dense_80_loss: 0.2986 - dense_81_loss: 0.2522 - dense_82_loss: 0.2422 - dense_63_acc: 0.9443 - dense_64_acc:
0.9443 - dense_65_acc: 0.9259 - dense_66_acc: 0.9559 - dense_67_acc: 0.9319 - dense_68_acc: 0.9599 - dense_69_acc: 0.8
193 - dense_70_acc: 0.9315 - dense_71_acc: 0.8786 - dense_72_acc: 0.9700 - dense_73_acc: 0.9407 - dense_74_acc: 0.9083
- dense_75_acc: 0.9323 - dense_76_acc: 0.9379 - dense_77_acc: 0.5493 - dense_78_acc: 0.9255 - dense_79_acc: 0.9768 - d
ense_80_acc: 0.9135 - dense_81_acc: 0.9427 - dense_82_acc: 0.9355 - val_loss: 6.5544 - val_dense_63_loss: 0.2884 - val
_dense_64_loss: 0.3143 - val_dense_65_loss: 0.3506 - val_dense_66_loss: 0.3046 - val_dense_67_loss: 0.2954 - val_dense
_68_loss: 0.2691 - val_dense_69_loss: 0.4473 - val_dense_70_loss: 0.2871 - val_dense_71_loss: 0.4329 - val_dense_72_lo
ss: 0.2162 - val_dense_73_loss: 0.3005 - val_dense_74_loss: 0.3664 - val_dense_75_loss: 0.2918 - val_dense_76_loss: 0.
2925 - val_dense_77_loss: 0.6758 - val_dense_78_loss: 0.2786 - val_dense_79_loss: 0.2098 - val_dense_80_loss: 0.3428 -
val_dense_81_loss: 0.3053 - val_dense_82_loss: 0.2849 - val_dense_63_acc: 0.9499 - val_dense_64_acc: 0.9467 - val_dens
e_65_acc: 0.9399 - val_dense_66_acc: 0.9603 - val_dense_67_acc: 0.9563 - val_dense_68_acc: 0.9611 - val_dense_69_acc:
0.8566 - val_dense_70_acc: 0.9291 - val_dense_71_acc: 0.8842 - val_dense_72_acc: 0.9700 - val_dense_73_acc: 0.9471 - v
al_dense_74_acc: 0.9119 - val_dense_75_acc: 0.9407 - val_dense_76_acc: 0.9499 - val_dense_77_acc: 0.5921 - val_dense_7
8_acc: 0.9523 - val_dense_79_acc: 0.9808 - val_dense_80_acc: 0.9267 - val_dense_81_acc: 0.9459 - val_dense_82_acc: 0.9
463
Epoch 2/10
78/78 [==============================] - 28s 363ms/step - loss: 5.3976 - dense_63_loss: 0.1883 - dense_64_loss: 0.2230
- dense_65_loss: 0.2776 - dense_66_loss: 0.1640 - dense_67_loss: 0.2478 - dense_68_loss: 0.1896 - dense_69_loss: 0.485
8 - dense_70_loss: 0.2477 - dense_71_loss: 0.3634 - dense_72_loss: 0.1443 - dense_73_loss: 0.2194 - dense_74_loss: 0.3
141 - dense_75_loss: 0.2544 - dense_76_loss: 0.2243 - dense_77_loss: 0.7411 - dense_78_loss: 0.2595 - dense_79_loss:
0.1059 - dense_80_loss: 0.2870 - dense_81_loss: 0.2303 - dense_82_loss: 0.2300 - dense_63_acc: 0.9555 - dense_64_acc:
0.9515 - dense_65_acc: 0.9267 - dense_66_acc: 0.9655 - dense_67_acc: 0.9370 - dense_68_acc: 0.9595 - dense_69_acc: 0.8
324 - dense_70_acc: 0.9339 - dense_71_acc: 0.8862 - dense_72_acc: 0.9724 - dense_73_acc: 0.9491 - dense_74_acc: 0.9108
- dense_75_acc: 0.9435 - dense_76_acc: 0.9519 - dense_77_acc: 0.5441 - dense_78_acc: 0.9375 - dense_79_acc: 0.9812 - d
ense_80_acc: 0.9192 - dense_81_acc: 0.9495 - dense_82_acc: 0.9423 - val_loss: 6.3714 - val_dense_63_loss: 0.2542 - val
_dense_64_loss: 0.2875 - val_dense_65_loss: 0.3203 - val_dense_66_loss: 0.2383 - val_dense_67_loss: 0.2955 - val_dense
_68_loss: 0.2680 - val_dense_69_loss: 0.4940 - val_dense_70_loss: 0.2958 - val_dense_71_loss: 0.3977 - val_dense_72_lo
ss: 0.2229 - val_dense_73_loss: 0.2834 - val_dense_74_loss: 0.3808 - val_dense_75_loss: 0.3047 - val_dense_76_loss: 0.
2792 - val_dense_77_loss: 0.6696 - val_dense_78_loss: 0.2939 - val_dense_79_loss: 0.1727 - val_dense_80_loss: 0.3547 -
val_dense_81_loss: 0.2651 - val_dense_82_loss: 0.2930 - val_dense_63_acc: 0.9487 - val_dense_64_acc: 0.9471 - val_dens
e_65_acc: 0.9395 - val_dense_66_acc: 0.9596 - val_dense_67_acc: 0.9564 - val_dense_68_acc: 0.9617 - val_dense_69_acc:
0.8563 - val_dense_70_acc: 0.9302 - val_dense_71_acc: 0.8842 - val_dense_72_acc: 0.9705 - val_dense_73_acc: 0.9459 - v
al_dense_74_acc: 0.9120 - val_dense_75_acc: 0.9395 - val_dense_76_acc: 0.9504 - val_dense_77_acc: 0.5868 - val_dense_7
8_acc: 0.9520 - val_dense_79_acc: 0.9810 - val_dense_80_acc: 0.9262 - val_dense_81_acc: 0.9463 - val_dense_82_acc: 0.9
459
Epoch 3/10
78/78 [==============================] - 29s 370ms/step - loss: 5.1209 - dense_63_loss: 0.1645 - dense_64_loss: 0.2101
- dense_65_loss: 0.2600 - dense_66_loss: 0.1566 - dense_67_loss: 0.2343 - dense_68_loss: 0.1757 - dense_69_loss: 0.455
1 - dense_70_loss: 0.2479 - dense_71_loss: 0.3485 - dense_72_loss: 0.1357 - dense_73_loss: 0.2133 - dense_74_loss: 0.2
898 - dense_75_loss: 0.2383 - dense_76_loss: 0.2272 - dense_77_loss: 0.7142 - dense_78_loss: 0.2486 - dense_79_loss:
0.0996 - dense_80_loss: 0.2637 - dense_81_loss: 0.2098 - dense_82_loss: 0.2281 - dense_63_acc: 0.9535 - dense_64_acc:
0.9523 - dense_65_acc: 0.9287 - dense_66_acc: 0.9642 - dense_67_acc: 0.9387 - dense_68_acc: 0.9611 - dense_69_acc: 0.8
304 - dense_70_acc: 0.9339 - dense_71_acc: 0.8841 - dense_72_acc: 0.9724 - dense_73_acc: 0.9463 - dense_74_acc: 0.9195
- dense_75_acc: 0.9419 - dense_76_acc: 0.9456 - dense_77_acc: 0.5681 - dense_78_acc: 0.9403 - dense_79_acc: 0.9804 - d
ense_80_acc: 0.9271 - dense_81_acc: 0.9462 - dense_82_acc: 0.9427 - val_loss: 5.8003 - val_dense_63_loss: 0.2399 - val
_dense_64_loss: 0.2729 - val_dense_65_loss: 0.2937 - val_dense_66_loss: 0.2399 - val_dense_67_loss: 0.2409 - val_dense
_68_loss: 0.2179 - val_dense_69_loss: 0.4324 - val_dense_70_loss: 0.2887 - val_dense_71_loss: 0.3663 - val_dense_72_lo
ss: 0.1985 - val_dense_73_loss: 0.2630 - val_dense_74_loss: 0.3082 - val_dense_75_loss: 0.2745 - val_dense_76_loss: 0.
2626 - val_dense_77_loss: 0.6554 - val_dense_78_loss: 0.2634 - val_dense_79_loss: 0.1590 - val_dense_80_loss: 0.2994 -
val_dense_81_loss: 0.2507 - val_dense_82_loss: 0.2731 - val_dense_63_acc: 0.9516 - val_dense_64_acc: 0.9471 - val_dens
e_65_acc: 0.9395 - val_dense_66_acc: 0.9596 - val_dense_67_acc: 0.9564 - val_dense_68_acc: 0.9613 - val_dense_69_acc:
0.8571 - val_dense_70_acc: 0.9278 - val_dense_71_acc: 0.8850 - val_dense_72_acc: 0.9693 - val_dense_73_acc: 0.9467 - v
al_dense_74_acc: 0.9136 - val_dense_75_acc: 0.9427 - val_dense_76_acc: 0.9483 - val_dense_77_acc: 0.6146 - val_dense_7
8_acc: 0.9516 - val_dense_79_acc: 0.9810 - val_dense_80_acc: 0.9286 - val_dense_81_acc: 0.9459 - val_dense_82_acc: 0.9
451
Epoch 4/10
78/78 [==============================] - 27s 351ms/step - loss: 4.9497 - dense_63_loss: 0.1534 - dense_64_loss: 0.2148
- dense_65_loss: 0.2586 - dense_66_loss: 0.1464 - dense_67_loss: 0.2249 - dense_68_loss: 0.1769 - dense_69_loss: 0.421
0 - dense_70_loss: 0.2289 - dense_71_loss: 0.3530 - dense_72_loss: 0.1332 - dense_73_loss: 0.2001 - dense_74_loss: 0.2
994 - dense_75_loss: 0.2245 - dense_76_loss: 0.2056 - dense_77_loss: 0.6815 - dense_78_loss: 0.2519 - dense_79_loss:
0.0971 - dense_80_loss: 0.2657 - dense_81_loss: 0.2016 - dense_82_loss: 0.2112 - dense_63_acc: 0.9539 - dense_64_acc:
0.9490 - dense_65_acc: 0.9275 - dense_66_acc: 0.9651 - dense_67_acc: 0.9403 - dense_68_acc: 0.9578 - dense_69_acc: 0.8
357 - dense_70_acc: 0.9343 - dense_71_acc: 0.8829 - dense_72_acc: 0.9720 - dense_73_acc: 0.9495 - dense_74_acc: 0.9139
- dense_75_acc: 0.9426 - dense_76_acc: 0.9507 - dense_77_acc: 0.5982 - dense_78_acc: 0.9375 - dense_79_acc: 0.9796 - d
ense_80_acc: 0.9227 - dense_81_acc: 0.9462 - dense_82_acc: 0.9451 - val_loss: 5.5991 - val_dense_63_loss: 0.1944 - val
_dense_64_loss: 0.2507 - val_dense_65_loss: 0.2761 - val_dense_66_loss: 0.1973 - val_dense_67_loss: 0.2440 - val_dense
_68_loss: 0.2382 - val_dense_69_loss: 0.4102 - val_dense_70_loss: 0.2724 - val_dense_71_loss: 0.3767 - val_dense_72_lo
ss: 0.1943 - val_dense_73_loss: 0.2493 - val_dense_74_loss: 0.3200 - val_dense_75_loss: 0.2597 - val_dense_76_loss: 0.
2294 - val_dense_77_loss: 0.6624 - val_dense_78_loss: 0.2564 - val_dense_79_loss: 0.1460 - val_dense_80_loss: 0.3005 -
val_dense_81_loss: 0.2546 - val_dense_82_loss: 0.2664 - val_dense_63_acc: 0.9536 - val_dense_64_acc: 0.9479 - val_dens
e_65_acc: 0.9407 - val_dense_66_acc: 0.9596 - val_dense_67_acc: 0.9580 - val_dense_68_acc: 0.9613 - val_dense_69_acc:
0.8584 - val_dense_70_acc: 0.9294 - val_dense_71_acc: 0.8846 - val_dense_72_acc: 0.9705 - val_dense_73_acc: 0.9483 - v
al_dense_74_acc: 0.9120 - val_dense_75_acc: 0.9395 - val_dense_76_acc: 0.9500 - val_dense_77_acc: 0.6146 - val_dense_7
8_acc: 0.9520 - val_dense_79_acc: 0.9802 - val_dense_80_acc: 0.9241 - val_dense_81_acc: 0.9483 - val_dense_82_acc: 0.9
475
Epoch 5/10
78/78 [==============================] - 28s 353ms/step - loss: 4.8054 - dense_63_loss: 0.1563 - dense_64_loss: 0.1935
- dense_65_loss: 0.2538 - dense_66_loss: 0.1367 - dense_67_loss: 0.2206 - dense_68_loss: 0.1682 - dense_69_loss: 0.423
5 - dense_70_loss: 0.2375 - dense_71_loss: 0.3299 - dense_72_loss: 0.1384 - dense_73_loss: 0.1940 - dense_74_loss: 0.3
063 - dense_75_loss: 0.2110 - dense_76_loss: 0.1824 - dense_77_loss: 0.6793 - dense_78_loss: 0.2334 - dense_79_loss:
0.0939 - dense_80_loss: 0.2528 - dense_81_loss: 0.1714 - dense_82_loss: 0.2227 - dense_63_acc: 0.9522 - dense_64_acc:
0.9531 - dense_65_acc: 0.9271 - dense_66_acc: 0.9655 - dense_67_acc: 0.9383 - dense_68_acc: 0.9586 - dense_69_acc: 0.8
344 - dense_70_acc: 0.9317 - dense_71_acc: 0.8870 - dense_72_acc: 0.9711 - dense_73_acc: 0.9487 - dense_74_acc: 0.9137
- dense_75_acc: 0.9439 - dense_76_acc: 0.9543 - dense_77_acc: 0.5946 - dense_78_acc: 0.9407 - dense_79_acc: 0.9816 - d
ense_80_acc: 0.9263 - dense_81_acc: 0.9495 - dense_82_acc: 0.9407 - val_loss: 5.4996 - val_dense_63_loss: 0.2171 - val
_dense_64_loss: 0.2545 - val_dense_65_loss: 0.2768 - val_dense_66_loss: 0.1998 - val_dense_67_loss: 0.2371 - val_dense
_68_loss: 0.2070 - val_dense_69_loss: 0.4093 - val_dense_70_loss: 0.2639 - val_dense_71_loss: 0.3652 - val_dense_72_lo
ss: 0.1829 - val_dense_73_loss: 0.2581 - val_dense_74_loss: 0.3220 - val_dense_75_loss: 0.2548 - val_dense_76_loss: 0.
2260 - val_dense_77_loss: 0.6482 - val_dense_78_loss: 0.2597 - val_dense_79_loss: 0.1405 - val_dense_80_loss: 0.2787 -
val_dense_81_loss: 0.2426 - val_dense_82_loss: 0.2554 - val_dense_63_acc: 0.9520 - val_dense_64_acc: 0.9459 - val_dens
```

520
Epoch 9/10
78/78 [==============================] - 29s 370ms/step - loss: 4.4435 - dense_63_loss: 0.1260 - dense_64_loss: 0.1860 - dense_65_loss: 0.2196 - dense_66_loss: 0.1319 - dense_67_loss: 0.2034 - dense_68_loss: 0.1580 - dense_69_loss: 0.392 3 - dense_70_loss: 0.2120 - dense_71_loss: 0.3002 - dense_72_loss: 0.1258 - dense_73_loss: 0.1806 - dense_74_loss: 0.2 838 - dense_75_loss: 0.1983 - dense_76_loss: 0.1802 - dense_77_loss: 0.6610 - dense_78_loss: 0.2182 - dense_79_loss: 0.0721 - dense_80_loss: 0.2466 - dense_81_loss: 0.1509 - dense_82_loss: 0.1965 - dense_63_acc: 0.9603 - dense_64_acc: 0.9503 - dense_65_acc: 0.9347 - dense_66_acc: 0.9651 - dense_67_acc: 0.9415 - dense_68_acc: 0.9587 - dense_69_acc: 0.8 432 - dense_70_acc: 0.9301 - dense_71_acc: 0.8890 - dense_72_acc: 0.9691 - dense_73_acc: 0.9470 - dense_74_acc: 0.9145 - dense_75_acc: 0.9403 - dense_76_acc: 0.9462 - dense_77_acc: 0.6143 - dense_78_acc: 0.9399 - dense_79_acc: 0.9844 - d ense_80_acc: 0.9239 - dense_81_acc: 0.9503 - dense_82_acc: 0.9407 - val_loss: 4.8744 - val_dense_63_loss: 0.1395 - val _dense_64_loss: 0.2185 - val_dense_65_loss: 0.2133 - val_dense_66_loss: 0.1433 - val_dense_67_loss: 0.2193 - val_dense _68_loss: 0.1616 - val_dense_69_loss: 0.4257 - val_dense_70_loss: 0.2311 - val_dense_71_loss: 0.3561 - val_dense_72_lo ss: 0.1361 - val_dense_73_loss: 0.2237 - val_dense_74_loss: 0.2894 - val_dense_75_loss: 0.2235 - val_dense_76_loss: 0. 2362 - val_dense_77_loss: 0.6587 - val_dense_78_loss: 0.2209 - val_dense_79_loss: 0.0959 - val_dense_80_loss: 0.2549 - val_dense_81_loss: 0.1954 - val_dense_82_loss: 0.2310 - val_dense_63_acc: 0.9556 - val_dense_64_acc: 0.9463 - val_dens e_65_acc: 0.9419 - val_dense_66_acc: 0.9609 - val_dense_67_acc: 0.9560 - val_dense_68_acc: 0.9613 - val_dense_69_acc: 0.8253 - val_dense_70_acc: 0.9290 - val_dense_71_acc: 0.8801 - val_dense_72_acc: 0.9732 - dense_73_acc: 0.9479 - v al_dense_74_acc: 0.9120 - val_dense_75_acc: 0.9358 - val_dense_76_acc: 0.9483 - val_dense_77_acc: 0.6094 - val_dense_7 8_acc: 0.9524 - val_dense_79_acc: 0.9831 - val_dense_80_acc: 0.9278 - val_dense_81_acc: 0.9492 - val_dense_82_acc: 0.9 431
Epoch 10/10
78/78 [==============================] - 29s 375ms/step - loss: 4.3798 - dense_63_loss: 0.1210 - dense_64_loss: 0.1792 - dense_65_loss: 0.2393 - dense_66_loss: 0.1459 - dense_67_loss: 0.2079 - dense_68_loss: 0.1532 - dense_69_loss: 0.374 5 - dense_70_loss: 0.2081 - dense_71_loss: 0.3029 - dense_72_loss: 0.1153 - dense_73_loss: 0.1679 - dense_74_loss: 0.2 729 - dense_75_loss: 0.1716 - dense_76_loss: 0.1609 - dense_77_loss: 0.6451 - dense_78_loss: 0.2311 - dense_79_loss: 0.0930 - dense_80_loss: 0.2355 - dense_81_loss: 0.1629 - dense_82_loss: 0.1916 - dense_63_acc: 0.9627 - dense_64_acc: 0.9506 - dense_65_acc: 0.9263 - dense_66_acc: 0.9626 - dense_67_acc: 0.9423 - dense_68_acc: 0.9570 - dense_69_acc: 0.8 492 - dense_70_acc: 0.9331 - dense_71_acc: 0.8898 - dense_72_acc: 0.9718 - dense_73_acc: 0.9491 - dense_74_acc: 0.9143 - dense_75_acc: 0.9463 - dense_76_acc: 0.9531 - dense_77_acc: 0.6340 - dense_78_acc: 0.9362 - dense_79_acc: 0.9780 - d ense_80_acc: 0.9295 - dense_81_acc: 0.9491 - dense_82_acc: 0.9447 - val_loss: 4.7503 - val_dense_63_loss: 0.1410 - val _dense_64_loss: 0.2158 - val_dense_65_loss: 0.2448 - val_dense_66_loss: 0.1519 - val_dense_67_loss: 0.2070 - val_dense

```
_68_loss: 0.1588 - val_dense_69_loss: 0.3617 - val_dense_70_loss: 0.2331 - val_dense_71_loss: 0.3577 - val_dense_72_lo
ss: 0.1477 - val_dense_73_loss: 0.2090 - val_dense_74_loss: 0.2847 - val_dense_75_loss: 0.2008 - val_dense_76_loss: 0.
1823 - val_dense_77_loss: 0.6493 - val_dense_78_loss: 0.2272 - val_dense_79_loss: 0.1062 - val_dense_80_loss: 0.2572 -
val_dense_81_loss: 0.1994 - val_dense_82_loss: 0.2148 - val_dense_63_acc: 0.9544 - val_dense_64_acc: 0.9479 - val_dens
e_65_acc: 0.9350 - val_dense_66_acc: 0.9584 - val_dense_67_acc: 0.9588 - val_dense_68_acc: 0.9613 - val_dense_69_acc:
0.8660 - val_dense_70_acc: 0.9286 - val_dense_71_acc: 0.8818 - val_dense_72_acc: 0.9677 - val_dense_73_acc: 0.9447 - v
al_dense_74_acc: 0.9144 - val_dense_75_acc: 0.9443 - val_dense_76_acc: 0.9504 - val_dense_77_acc: 0.6203 - val_dense_7
8_acc: 0.9512 - val_dense_79_acc: 0.9806 - val_dense_80_acc: 0.9221 - val_dense_81_acc: 0.9447 - val_dense_82_acc: 0.9
443
```

Out[364]: `<keras.callbacks.History at 0x7f23837d4ba8>`

In [365]:
```
test_generator.reset()
pred_model3=model.predict_generator(test_generator,
                                    steps=STEP_SIZE_TEST,
                                    verbose=1)
```

```
4952/4952 [==============================] - 74s 15ms/step
```

In [366]:
```
res3 = np.array(pred_model3)
pred_three = []
for m in range(len(test_generator)):
    prob = []
    for c in range(len(columns)):
        prob.append(res3[c][m][0])
    pred_three.append(prob)

pred_three = np.array(pred_three)

pred_bool = (pred_three >0.5)


predictions = pred_bool.astype(int)
#columns should be the same order of y_col
results = pd.DataFrame(predictions, columns=columns)
results["Filenames"] = test_generator.filenames
ordered_cols = ["Filenames"]+columns
results = results[ordered_cols]#To get the same column order
#results[results.bird==1]
results[:10]
```

Out[366]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	007157.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	002736.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
2	003624.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
3	002106.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
4	002927.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	008964.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	004589.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
7	001698.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	008567.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	005119.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

In [367]:
```
#Saveing to harddrive
result = results
label = imageMap_test

prediction = result
actual = label

prediction = prediction.sort_values(by=['Filenames']).reset_index(drop=True)
actual = actual.sort_values(by=['Filenames']).reset_index(drop=True)

files = list(actual['Filenames'])
del prediction['Filenames']
del actual['Filenames']
pred_array = prediction.values
actual_array = actual.values
print('Total number of test samples: ',len(actual_array))

space = range(5,21)
for s in space:
    frac = int((s/20)*100)
    number = 0
    for n in range(len(actual_array)):
      if(((actual_array[n] == pred_array[n]).sum())>=s): number = number+1
    percent = number/len(actual_array)*100
    print('atleast ', frac,'% acurate predictions(',s,' out of 20) :',number,'  percentage: ',percent,'%')
```

```
Total number of test samples:  4952
atleast  25 % acurate predictions( 5  out of 20) : 4952   percentage:  100.0 %
atleast  30 % acurate predictions( 6  out of 20) : 4952   percentage:  100.0 %
atleast  35 % acurate predictions( 7  out of 20) : 4952   percentage:  100.0 %
atleast  40 % acurate predictions( 8  out of 20) : 4952   percentage:  100.0 %
atleast  45 % acurate predictions( 9  out of 20) : 4952   percentage:  100.0 %
atleast  50 % acurate predictions( 10  out of 20) : 4952   percentage:  100.0 %
atleast  55 % acurate predictions( 11  out of 20) : 4952   percentage:  100.0 %
atleast  60 % acurate predictions( 12  out of 20) : 4952   percentage:  100.0 %
atleast  65 % acurate predictions( 13  out of 20) : 4952   percentage:  100.0 %
atleast  70 % acurate predictions( 14  out of 20) : 4952   percentage:  100.0 %
atleast  75 % acurate predictions( 15  out of 20) : 4952   percentage:  100.0 %
atleast  80 % acurate predictions( 16  out of 20) : 4944   percentage:  99.83844911147011 %
atleast  85 % acurate predictions( 17  out of 20) : 4878   percentage:  98.50565428109854 %
atleast  90 % acurate predictions( 18  out of 20) : 4460   percentage:  90.06462035541195 %
atleast  95 % acurate predictions( 19  out of 20) : 2613   percentage:  52.766558966074314 %
atleast  100 % acurate predictions( 20  out of 20) : 391   percentage:  7.895799676898223 %
```

In [386]:
```
correct_pred = []
for n in range(len(actual_array)):
      if(((actual_array[n] == pred_array[n]).sum())>19): correct_pred.append(files[n])
correct_pred[20:25]
```

Out[386]:
```
['000502.jpg', '000521.jpg', '000534.jpg', '000566.jpg', '000624.jpg']
```

In [388]:
```
#Lets see some examples of correct predictions
print(result[result['Filenames']=='000624.jpg'].to_string())
Image("/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images/000624.jpg")
```

```
      Filenames  aeroplane  bicycle  bird  boat  bottle  bus  car  cat  chair  cow  diningtable  dog  horse  motorbik
e  person  pottedplant  sheep  sofa  train  tvmonitor
2263  000624.jpg          0        0     0     0       0    0    1    0      0    0            0    0      0
0       1           0      0     0     0          0
```

Out[388]:

In [391]:
```
correct_pred[225:230]
```

Out[391]:
```
['005748.jpg', '005842.jpg', '005865.jpg', '005887.jpg', '005900.jpg']
```

In [396]:
```
#Lets see some examples of correct predictions
print(result[result['Filenames']=='005842.jpg'].to_string())
Image("/content/drive/My Drive/Assignment6/Data/objectDetectionDb/Test/Images/005842.jpg")
```

```
      Filenames  aeroplane  bicycle  bird  boat  bottle  bus  car  cat  chair  cow  diningtable  dog  horse  motorbik
e  person  pottedplant  sheep  sofa  train  tvmonitor
1874  005842.jpg          0        0     0     0       0    0    1    0      0    0            0    0      0
0       1            0      0     0      0          0
```

Out[396]:

Version2

In [398]:
```
train_generator, valid_generator, test_generator = generate_modified_db2(imageMap_trainval, imageMap_test, columns)
```

```
For Training:
Found 4811 images.
For Validation:
Found 200 images.
For Testing:
Found 4952 images.
```

In [0]:
```
inp = Input(shape = (100,100,3))
x = Conv2D(32, (3, 3), padding = 'same')(inp)
x = Activation('relu')(x)
x = Conv2D(32, (3, 3))(x)
x = Activation('relu')(x)
x = MaxPooling2D(pool_size = (2, 2))(x)
x = Dropout(0.25)(x)
x = Conv2D(64, (3, 3), padding = 'same')(x)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3))(x)
x = Activation('relu')(x)
x = MaxPooling2D(pool_size = (2, 2))(x)
x = Dropout(0.25)(x)
x = Flatten()(x)
x = Dense(512)(x)
x = Activation('relu')(x)
x = Dropout(0.5)(x)
output1 = Dense(1, activation = 'sigmoid')(x)
output2 = Dense(1, activation = 'sigmoid')(x)
output3 = Dense(1, activation = 'sigmoid')(x)
output4 = Dense(1, activation = 'sigmoid')(x)
output5 = Dense(1, activation = 'sigmoid')(x)
output6 = Dense(1, activation = 'sigmoid')(x)
output7 = Dense(1, activation = 'sigmoid')(x)
output8 = Dense(1, activation = 'sigmoid')(x)
output9 = Dense(1, activation = 'sigmoid')(x)
output10 = Dense(1, activation = 'sigmoid')(x)
output11 = Dense(1, activation = 'sigmoid')(x)
output12 = Dense(1, activation = 'sigmoid')(x)
output13 = Dense(1, activation = 'sigmoid')(x)
output14 = Dense(1, activation = 'sigmoid')(x)
output15 = Dense(1, activation = 'sigmoid')(x)
output16 = Dense(1, activation = 'sigmoid')(x)
output17 = Dense(1, activation = 'sigmoid')(x)
output18 = Dense(1, activation = 'sigmoid')(x)
output19 = Dense(1, activation = 'sigmoid')(x)
output20 = Dense(1, activation = 'sigmoid')(x)
model = Model(inp,[output1,output2,output3,output4,output5,output6,output7,
                   output8,output9,output10,output11,output12,output13,output14,
                   output15,output16,output17,output18,output19,output20])
model.compile(optimizers.rmsprop(lr = 0.0001, decay = 1e-6),
loss = ["binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy",
        "binary_crossentropy","binary_crossentropy"],metrics = ["accuracy"])
```

In [0]:
```
def generator_wrapper(generator):
    for batch_x,batch_y in generator:
        yield (batch_x,[batch_y[:,i] for i in range(20)])
```

In [402]:
```
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size
model.fit_generator(generator = generator_wrapper(train_generator),
                    steps_per_epoch = STEP_SIZE_TRAIN,
                    validation_data = generator_wrapper(valid_generator),
                    validation_steps = STEP_SIZE_VALID,
                    epochs=10,verbose=1
                   )
```

```
Epoch 1/10
150/150 [==============================] - 37s 246ms/step - loss: 5.5323 - dense_105_loss: 0.2060 - dense_106_loss: 0.
2483 - dense_107_loss: 0.2787 - dense_108_loss: 0.1827 - dense_109_loss: 0.2353 - dense_110_loss: 0.2004 - dense_111_l
oss: 0.4742 - dense_112_loss: 0.2641 - dense_113_loss: 0.3806 - dense_114_loss: 0.1641 - dense_115_loss: 0.2319 - dens
e_116_loss: 0.3139 - dense_117_loss: 0.2488 - dense_118_loss: 0.2328 - dense_119_loss: 0.7452 - dense_120_loss: 0.2509
- dense_121_loss: 0.1210 - dense_122_loss: 0.2832 - dense_123_loss: 0.2361 - dense_124_loss: 0.2342 - dense_105_acc:
0.9481 - dense_106_acc: 0.9456 - dense_107_acc: 0.9267 - dense_108_acc: 0.9613 - dense_109_acc: 0.9440 - dense_110_ac
c: 0.9575 - dense_111_acc: 0.8344 - dense_112_acc: 0.9285 - dense_113_acc: 0.8829 - dense_114_acc: 0.9667 - dense_115_
acc: 0.9431 - dense_116_acc: 0.9121 - dense_117_acc: 0.9421 - dense_118_acc: 0.9452 - dense_119_acc: 0.5400 - dense_12
0_acc: 0.9442 - dense_121_acc: 0.9752 - dense_122_acc: 0.9206 - dense_123_acc: 0.9419 - dense_124_acc: 0.9406 - val_lo
ss: 5.8262 - val_dense_105_loss: 0.2269 - val_dense_106_loss: 0.2655 - val_dense_107_loss: 0.1986 - val_dense_108_los
s: 0.2277 - val_dense_109_loss: 0.2101 - val_dense_110_loss: 0.2054 - val_dense_111_loss: 0.4728 - val_dense_112_loss:
0.2875 - val_dense_113_loss: 0.4195 - val_dense_114_loss: 0.1577 - val_dense_115_loss: 0.2406 - val_dense_116_loss: 0.
3305 - val_dense_117_loss: 0.3161 - val_dense_118_loss: 0.2612 - val_dense_119_loss: 0.6469 - val_dense_120_loss: 0.30
19 - val_dense_121_loss: 0.1665 - val_dense_122_loss: 0.3234 - val_dense_123_loss: 0.2414 - val_dense_124_loss: 0.3262
- val_dense_105_acc: 0.9479 - val_dense_106_acc: 0.9427 - val_dense_107_acc: 0.9635 - val_dense_108_acc: 0.9479 - val_
dense_109_acc: 0.9792 - val_dense_110_acc: 0.9740 - val_dense_111_acc: 0.8229 - val_dense_112_acc: 0.9479 - val_dense_
113_acc: 0.8698 - val_dense_114_acc: 0.9896 - val_dense_115_acc: 0.9635 - val_dense_116_acc: 0.9167 - val_dense_117_ac
c: 0.9115 - val_dense_118_acc: 0.9479 - val_dense_119_acc: 0.6510 - val_dense_120_acc: 0.9271 - val_dense_121_acc: 0.9
740 - val_dense_122_acc: 0.9167 - val_dense_123_acc: 0.9583 - val_dense_124_acc: 0.9167
Epoch 2/10
150/150 [==============================] - 30s 199ms/step - loss: 5.0632 - dense_105_loss: 0.1643 - dense_106_loss: 0.
2235 - dense_107_loss: 0.2520 - dense_108_loss: 0.1515 - dense_109_loss: 0.2177 - dense_110_loss: 0.1790 - dense_111_l
oss: 0.4352 - dense_112_loss: 0.2481 - dense_113_loss: 0.3571 - dense_114_loss: 0.1451 - dense_115_loss: 0.2169 - dens
e_116_loss: 0.2996 - dense_117_loss: 0.2270 - dense_118_loss: 0.2215 - dense_119_loss: 0.6994 - dense_120_loss: 0.2216
- dense_121_loss: 0.1023 - dense_122_loss: 0.2674 - dense_123_loss: 0.2108 - dense_124_loss: 0.2233 - dense_105_acc:
0.9527 - dense_106_acc: 0.9490 - dense_107_acc: 0.9321 - dense_108_acc: 0.9633 - dense_109_acc: 0.9461 - dense_110_ac
c: 0.9606 - dense_111_acc: 0.8459 - dense_112_acc: 0.9312 - dense_113_acc: 0.8846 - dense_114_acc: 0.9700 - dense_115_
acc: 0.9452 - dense_116_acc: 0.9146 - dense_117_acc: 0.9423 - dense_118_acc: 0.9496 - dense_119_acc: 0.5674 - dense_12
0_acc: 0.9459 - dense_121_acc: 0.9810 - dense_122_acc: 0.9252 - dense_123_acc: 0.9465 - dense_124_acc: 0.9454 - val_lo
ss: 5.7689 - val_dense_105_loss: 0.2363 - val_dense_106_loss: 0.2929 - val_dense_107_loss: 0.2704 - val_dense_108_los
s: 0.2436 - val_dense_109_loss: 0.1997 - val_dense_110_loss: 0.2655 - val_dense_111_loss: 0.5113 - val_dense_112_loss:
0.2521 - val_dense_113_loss: 0.3715 - val_dense_114_loss: 0.1984 - val_dense_115_loss: 0.2009 - val_dense_116_loss: 0.
3104 - val_dense_117_loss: 0.3345 - val_dense_118_loss: 0.2451 - val_dense_119_loss: 0.6389 - val_dense_120_loss: 0.26
26 - val_dense_121_loss: 0.1815 - val_dense_122_loss: 0.2562 - val_dense_123_loss: 0.2360 - val_dense_124_loss: 0.2611
- val_dense_105_acc: 0.9524 - val_dense_106_acc: 0.9524 - val_dense_107_acc: 0.9583 - val_dense_108_acc: 0.9524 - val_
dense_109_acc: 0.9762 - val_dense_110_acc: 0.9464 - val_dense_111_acc: 0.8095 - val_dense_112_acc: 0.9464 - val_dense_
113_acc: 0.8810 - val_dense_114_acc: 0.9762 - val_dense_115_acc: 0.9643 - val_dense_116_acc: 0.9107 - val_dense_117_ac
c: 0.8988 - val_dense_118_acc: 0.9524 - val_dense_119_acc: 0.6667 - val_dense_120_acc: 0.9464 - val_dense_121_acc: 0.9
643 - val_dense_122_acc: 0.9345 - val_dense_123_acc: 0.9583 - val_dense_124_acc: 0.9345
Epoch 3/10
150/150 [==============================] - 32s 214ms/step - loss: 4.8147 - dense_105_loss: 0.1527 - dense_106_loss: 0.
2111 - dense_107_loss: 0.2400 - dense_108_loss: 0.1454 - dense_109_loss: 0.2154 - dense_110_loss: 0.1775 - dense_111_l
oss: 0.4187 - dense_112_loss: 0.2383 - dense_113_loss: 0.3436 - dense_114_loss: 0.1345 - dense_115_loss: 0.1957 - dens
e_116_loss: 0.2869 - dense_117_loss: 0.2067 - dense_118_loss: 0.1965 - dense_119_loss: 0.6844 - dense_120_loss: 0.2156
- dense_121_loss: 0.0940 - dense_122_loss: 0.2572 - dense_123_loss: 0.1860 - dense_124_loss: 0.2144 - dense_105_acc:
0.9521 - dense_106_acc: 0.9492 - dense_107_acc: 0.9323 - dense_108_acc: 0.9621 - dense_109_acc: 0.9450 - dense_110_ac
c: 0.9586 - dense_111_acc: 0.8401 - dense_112_acc: 0.9306 - dense_113_acc: 0.8854 - dense_114_acc: 0.9700 - dense_115_
acc: 0.9475 - dense_116_acc: 0.9160 - dense_117_acc: 0.9429 - dense_118_acc: 0.9504 - dense_119_acc: 0.5926 - dense_12
0_acc: 0.9467 - dense_121_acc: 0.9815 - dense_122_acc: 0.9252 - dense_123_acc: 0.9479 - dense_124_acc: 0.9465 - val_lo
ss: 5.1717 - val_dense_105_loss: 0.1767 - val_dense_106_loss: 0.2712 - val_dense_107_loss: 0.1859 - val_dense_108_los
s: 0.1816 - val_dense_109_loss: 0.1666 - val_dense_110_loss: 0.1815 - val_dense_111_loss: 0.4498 - val_dense_112_loss:
0.2211 - val_dense_113_loss: 0.3631 - val_dense_114_loss: 0.1290 - val_dense_115_loss: 0.1903 - val_dense_116_loss: 0.
2900 - val_dense_117_loss: 0.3030 - val_dense_118_loss: 0.2337 - val_dense_119_loss: 0.6317 - val_dense_120_loss: 0.28
64 - val_dense_121_loss: 0.1357 - val_dense_122_loss: 0.2742 - val_dense_123_loss: 0.2169 - val_dense_124_loss: 0.2832
- val_dense_105_acc: 0.9524 - val_dense_106_acc: 0.9286 - val_dense_107_acc: 0.9702 - val_dense_108_acc: 0.9464 - val_
dense_109_acc: 0.9821 - val_dense_110_acc: 0.9702 - val_dense_111_acc: 0.8631 - val_dense_112_acc: 0.9464 - val_dense_
113_acc: 0.8750 - val_dense_114_acc: 0.9821 - val_dense_115_acc: 0.9702 - val_dense_116_acc: 0.9167 - val_dense_117_ac
c: 0.8988 - val_dense_118_acc: 0.9405 - val_dense_119_acc: 0.6250 - val_dense_120_acc: 0.9226 - val_dense_121_acc: 0.9
643 - val_dense_122_acc: 0.9107 - val_dense_123_acc: 0.9583 - val_dense_124_acc: 0.9167
Epoch 4/10
150/150 [==============================] - 31s 205ms/step - loss: 4.6363 - dense_105_loss: 0.1398 - dense_106_loss: 0.
1933 - dense_107_loss: 0.2299 - dense_108_loss: 0.1363 - dense_109_loss: 0.2060 - dense_110_loss: 0.1629 - dense_111_l
oss: 0.3983 - dense_112_loss: 0.2294 - dense_113_loss: 0.3260 - dense_114_loss: 0.1343 - dense_115_loss: 0.1951 - dens
e_116_loss: 0.2903 - dense_117_loss: 0.1962 - dense_118_loss: 0.1836 - dense_119_loss: 0.6796 - dense_120_loss: 0.2095
- dense_121_loss: 0.0863 - dense_122_loss: 0.2478 - dense_123_loss: 0.1806 - dense_124_loss: 0.2108 - dense_105_acc:
0.9550 - dense_106_acc: 0.9494 - dense_107_acc: 0.9315 - dense_108_acc: 0.9627 - dense_109_acc: 0.9465 - dense_110_ac
c: 0.9606 - dense_111_acc: 0.8490 - dense_112_acc: 0.9302 - dense_113_acc: 0.8863 - dense_114_acc: 0.9696 - dense_115_
acc: 0.9460 - dense_116_acc: 0.9127 - dense_117_acc: 0.9431 - dense_118_acc: 0.9512 - dense_119_acc: 0.6123 - dense_12
0_acc: 0.9463 - dense_121_acc: 0.9815 - dense_122_acc: 0.9254 - dense_123_acc: 0.9454 - dense_124_acc: 0.9436 - val_lo
ss: 4.7047 - val_dense_105_loss: 0.1623 - val_dense_106_loss: 0.2261 - val_dense_107_loss: 0.1516 - val_dense_108_los
s: 0.1634 - val_dense_109_loss: 0.1532 - val_dense_110_loss: 0.1395 - val_dense_111_loss: 0.4491 - val_dense_112_loss:
0.2223 - val_dense_113_loss: 0.3664 - val_dense_114_loss: 0.1004 - val_dense_115_loss: 0.1630 - val_dense_116_loss: 0.
3167 - val_dense_117_loss: 0.2774 - val_dense_118_loss: 0.1859 - val_dense_119_loss: 0.6123 - val_dense_120_loss: 0.19
78 - val_dense_121_loss: 0.1074 - val_dense_122_loss: 0.2707 - val_dense_123_loss: 0.1676 - val_dense_124_loss: 0.2717
- val_dense_105_acc: 0.9464 - val_dense_106_acc: 0.9524 - val_dense_107_acc: 0.9821 - val_dense_108_acc: 0.9524 - val_
dense_109_acc: 0.9821 - val_dense_110_acc: 0.9821 - val_dense_111_acc: 0.8036 - val_dense_112_acc: 0.9464 - val_dense_
113_acc: 0.8571 - val_dense_114_acc: 0.9940 - val_dense_115_acc: 0.9702 - val_dense_116_acc: 0.8869 - val_dense_117_ac
c: 0.8929 - val_dense_118_acc: 0.9643 - val_dense_119_acc: 0.6667 - val_dense_120_acc: 0.9524 - val_dense_121_acc: 0.9
762 - val_dense_122_acc: 0.9226 - val_dense_123_acc: 0.9524 - val_dense_124_acc: 0.9226
Epoch 5/10
150/150 [==============================] - 32s 211ms/step - loss: 4.5167 - dense_105_loss: 0.1451 - dense_106_loss: 0.
1907 - dense_107_loss: 0.2314 - dense_108_loss: 0.1400 - dense_109_loss: 0.2010 - dense_110_loss: 0.1587 - dense_111_l
oss: 0.3776 - dense_112_loss: 0.2299 - dense_113_loss: 0.3205 - dense_114_loss: 0.1261 - dense_115_loss: 0.1914 - dens
e_116_loss: 0.2736 - dense_117_loss: 0.1879 - dense_118_loss: 0.1820 - dense_119_loss: 0.6522 - dense_120_loss: 0.2078
- dense_121_loss: 0.0877 - dense_122_loss: 0.2300 - dense_123_loss: 0.1758 - dense_124_loss: 0.2072 - dense_105_acc:
0.9511 - dense_106_acc: 0.9500 - dense_107_acc: 0.9321 - dense_108_acc: 0.9609 - dense_109_acc: 0.9465 - dense_110_ac
c: 0.9606 - dense_111_acc: 0.8492 - dense_112_acc: 0.9302 - dense_113_acc: 0.8838 - dense_114_acc: 0.9715 - dense_115_
acc: 0.9461 - dense_116_acc: 0.9173 - dense_117_acc: 0.9425 - dense_118_acc: 0.9494 - dense_119_acc: 0.6278 - dense_12
0_acc: 0.9450 - dense_121_acc: 0.9806 - dense_122_acc: 0.9304 - dense_123_acc: 0.9467 - dense_124_acc: 0.9442 - val_lo
ss: 4.7684 - val_dense_105_loss: 0.1229 - val_dense_106_loss: 0.2340 - val_dense_107_loss: 0.1868 - val_dense_108_los
s: 0.1434 - val_dense_109_loss: 0.1360 - val_dense_110_loss: 0.1949 - val_dense_111_loss: 0.4827 - val_dense_112_loss:
0.1591 - val_dense_113_loss: 0.3478 - val_dense_114_loss: 0.1231 - val_dense_115_loss: 0.1683 - val_dense_116_loss: 0.
2688 - val_dense_117_loss: 0.2396 - val_dense_118_loss: 0.2156 - val_dense_119_loss: 0.6246 - val_dense_120_loss: 0.32
92 - val_dense_121_loss: 0.0971 - val_dense_122_loss: 0.2279 - val_dense_123_loss: 0.1784 - val_dense_124_loss: 0.2881
```

- val_dense_105_acc: 0.9643 - val_dense_106_acc: 0.9405 - val_dense_107_acc: 0.9524 - val_dense_108_acc: 0.9643 - val_dense_109_acc: 0.9821 - val_dense_110_acc: 0.9524 - val_dense_111_acc: 0.8036 - val_dense_112_acc: 0.9643 - val_dense_113_acc: 0.8810 - val_dense_114_acc: 0.9762 - val_dense_115_acc: 0.9643 - val_dense_116_acc: 0.9167 - val_dense_117_acc: 0.9345 - val_dense_118_acc: 0.9405 - val_dense_119_acc: 0.6310 - val_dense_120_acc: 0.8869 - val_dense_121_acc: 0.9702 - val_dense_122_acc: 0.9226 - val_dense_123_acc: 0.9524 - val_dense_124_acc: 0.9048

Epoch 6/10

150/150 [==============================] - 31s 209ms/step - loss: 4.4116 - dense_105_loss: 0.1332 - dense_106_loss: 0.1901 - dense_107_loss: 0.2277 - dense_108_loss: 0.1263 - dense_109_loss: 0.1961 - dense_110_loss: 0.1524 - dense_111_loss: 0.3672 - dense_112_loss: 0.2234 - dense_113_loss: 0.3139 - dense_114_loss: 0.1261 - dense_115_loss: 0.1834 - dense_116_loss: 0.2842 - dense_117_loss: 0.1842 - dense_118_loss: 0.1775 - dense_119_loss: 0.6406 - dense_120_loss: 0.1999 - dense_121_loss: 0.0856 - dense_122_loss: 0.2382 - dense_123_loss: 0.1681 - dense_124_loss: 0.1934 - dense_105_acc: 0.9542 - dense_106_acc: 0.9485 - dense_107_acc: 0.9319 - dense_108_acc: 0.9644 - dense_109_acc: 0.9465 - dense_110_acc: 0.9604 - dense_111_acc: 0.8571 - dense_112_acc: 0.9288 - dense_113_acc: 0.8852 - dense_114_acc: 0.9698 - dense_115_acc: 0.9462 - dense_116_acc: 0.9115 - dense_117_acc: 0.9430 - dense_118_acc: 0.9475 - dense_119_acc: 0.6335 - dense_120_acc: 0.9465 - dense_121_acc: 0.9806 - dense_122_acc: 0.9231 - dense_123_acc: 0.9475 - dense_124_acc: 0.9460 - val_loss: 4.7155 - val_dense_105_loss: 0.1660 - val_dense_106_loss: 0.2191 - val_dense_107_loss: 0.1352 - val_dense_108_loss: 0.1635 - val_dense_109_loss: 0.1573 - val_dense_110_loss: 0.1547 - val_dense_111_loss: 0.4036 - val_dense_112_loss: 0.2338 - val_dense_113_loss: 0.3880 - val_dense_114_loss: 0.0856 - val_dense_115_loss: 0.1818 - val_dense_116_loss: 0.3204 - val_dense_117_loss: 0.2768 - val_dense_118_loss: 0.1943 - val_dense_119_loss: 0.6249 - val_dense_120_loss: 0.2155 - val_dense_121_loss: 0.0852 - val_dense_122_loss: 0.2789 - val_dense_123_loss: 0.1500 - val_dense_124_loss: 0.2808 - val_dense_105_acc: 0.9643 - val_dense_106_acc: 0.9583 - val_dense_107_acc: 0.9643 - val_dense_108_acc: 0.9464 - val_dense_109_acc: 0.9762 - val_dense_110_acc: 0.9643 - val_dense_111_acc: 0.8571 - val_dense_112_acc: 0.9524 - val_dense_113_acc: 0.8631 - val_dense_114_acc: 0.9940 - val_dense_115_acc: 0.9583 - val_dense_116_acc: 0.9048 - val_dense_117_acc: 0.8929 - val_dense_118_acc: 0.9464 - val_dense_119_acc: 0.6726 - val_dense_120_acc: 0.9464 - val_dense_121_acc: 0.9762 - val_dense_122_acc: 0.9048 - val_dense_123_acc: 0.9583 - val_dense_124_acc: 0.9107

Epoch 7/10

150/150 [==============================] - 32s 216ms/step - loss: 4.3057 - dense_105_loss: 0.1286 - dense_106_loss: 0.1768 - dense_107_loss: 0.2217 - dense_108_loss: 0.1333 - dense_109_loss: 0.1938 - dense_110_loss: 0.1502 - dense_111_loss: 0.3538 - dense_112_loss: 0.2147 - dense_113_loss: 0.3030 - dense_114_loss: 0.1257 - dense_115_loss: 0.1791 - dense_116_loss: 0.2736 - dense_117_loss: 0.1791 - dense_118_loss: 0.1684 - dense_119_loss: 0.6350 - dense_120_loss: 0.1985 - dense_121_loss: 0.0794 - dense_122_loss: 0.2292 - dense_123_loss: 0.1703 - dense_124_loss: 0.1916 - dense_105_acc: 0.9537 - dense_106_acc: 0.9498 - dense_107_acc: 0.9325 - dense_108_acc: 0.9623 - dense_109_acc: 0.9458 - dense_110_acc: 0.9604 - dense_111_acc: 0.8581 - dense_112_acc: 0.9321 - dense_113_acc: 0.8861 - dense_114_acc: 0.9698 - dense_115_acc: 0.9446 - dense_116_acc: 0.9144 - dense_117_acc: 0.9456 - dense_118_acc: 0.9504 - dense_119_acc: 0.6442 - dense_120_acc: 0.9460 - dense_121_acc: 0.9815 - dense_122_acc: 0.9232 - dense_123_acc: 0.9444 - dense_124_acc: 0.9454 - val_loss: 4.2766 - val_dense_105_loss: 0.1524 - val_dense_106_loss: 0.2031 - val_dense_107_loss: 0.1631 - val_dense_108_loss: 0.1722 - val_dense_109_loss: 0.1094 - val_dense_110_loss: 0.1520 - val_dense_111_loss: 0.4224 - val_dense_112_loss: 0.1815 - val_dense_113_loss: 0.3219 - val_dense_114_loss: 0.1032 - val_dense_115_loss: 0.1320 - val_dense_116_loss: 0.2533 - val_dense_117_loss: 0.2380 - val_dense_118_loss: 0.1558 - val_dense_119_loss: 0.6099 - val_dense_120_loss: 0.2392 - val_dense_121_loss: 0.1027 - val_dense_122_loss: 0.2180 - val_dense_123_loss: 0.1214 - val_dense_124_loss: 0.2252 - val_dense_105_acc: 0.9464 - val_dense_106_acc: 0.9464 - val_dense_107_acc: 0.9643 - val_dense_108_acc: 0.9405 - val_dense_109_acc: 0.9821 - val_dense_110_acc: 0.9643 - val_dense_111_acc: 0.8274 - val_dense_112_acc: 0.9464 - val_dense_113_acc: 0.8750 - val_dense_114_acc: 0.9821 - val_dense_115_acc: 0.9643 - val_dense_116_acc: 0.9167 - val_dense_117_acc: 0.9107 - val_dense_118_acc: 0.9524 - val_dense_119_acc: 0.6905 - val_dense_120_acc: 0.9286 - val_dense_121_acc: 0.9643 - val_dense_122_acc: 0.9286 - val_dense_123_acc: 0.9583 - val_dense_124_acc: 0.9345

Epoch 8/10

150/150 [==============================] - 32s 217ms/step - loss: 4.2030 - dense_105_loss: 0.1227 - dense_106_loss: 0.1748 - dense_107_loss: 0.2200 - dense_108_loss: 0.1244 - dense_109_loss: 0.1919 - dense_110_loss: 0.1427 - dense_111_loss: 0.3533 - dense_112_loss: 0.2142 - dense_113_loss: 0.2952 - dense_114_loss: 0.1220 - dense_115_loss: 0.1698 - dense_116_loss: 0.2728 - dense_117_loss: 0.1717 - dense_118_loss: 0.1648 - dense_119_loss: 0.6242 - dense_120_loss: 0.1941 - dense_121_loss: 0.0791 - dense_122_loss: 0.2261 - dense_123_loss: 0.1559 - dense_124_loss: 0.1830 - dense_105_acc: 0.9560 - dense_106_acc: 0.9519 - dense_107_acc: 0.9306 - dense_108_acc: 0.9625 - dense_109_acc: 0.9457 - dense_110_acc: 0.9615 - dense_111_acc: 0.8573 - dense_112_acc: 0.9294 - dense_113_acc: 0.8904 - dense_114_acc: 0.9704 - dense_115_acc: 0.9475 - dense_116_acc: 0.9123 - dense_117_acc: 0.9467 - dense_118_acc: 0.9510 - dense_119_acc: 0.6507 - dense_120_acc: 0.9458 - dense_121_acc: 0.9810 - dense_122_acc: 0.9242 - dense_123_acc: 0.9506 - dense_124_acc: 0.9454 - val_loss: 4.3606 - val_dense_105_loss: 0.1302 - val_dense_106_loss: 0.2145 - val_dense_107_loss: 0.1420 - val_dense_108_loss: 0.1535 - val_dense_109_loss: 0.1213 - val_dense_110_loss: 0.1464 - val_dense_111_loss: 0.4200 - val_dense_112_loss: 0.1973 - val_dense_113_loss: 0.3480 - val_dense_114_loss: 0.0968 - val_dense_115_loss: 0.1564 - val_dense_116_loss: 0.2515 - val_dense_117_loss: 0.2439 - val_dense_118_loss: 0.1780 - val_dense_119_loss: 0.5996 - val_dense_120_loss: 0.2587 - val_dense_121_loss: 0.0890 - val_dense_122_loss: 0.2304 - val_dense_123_loss: 0.1423 - val_dense_124_loss: 0.2407 - val_dense_105_acc: 0.9583 - val_dense_106_acc: 0.9531 - val_dense_107_acc: 0.9635 - val_dense_108_acc: 0.9479 - val_dense_109_acc: 0.9792 - val_dense_110_acc: 0.9635 - val_dense_111_acc: 0.8281 - val_dense_112_acc: 0.9479 - val_dense_113_acc: 0.8646 - val_dense_114_acc: 0.9844 - val_dense_115_acc: 0.9635 - val_dense_116_acc: 0.9167 - val_dense_117_acc: 0.9115 - val_dense_118_acc: 0.9427 - val_dense_119_acc: 0.7031 - val_dense_120_acc: 0.9271 - val_dense_121_acc: 0.9688 - val_dense_122_acc: 0.9219 - val_dense_123_acc: 0.9583 - val_dense_124_acc: 0.9271

Epoch 9/10

150/150 [==============================] - 31s 205ms/step - loss: 4.0874 - dense_105_loss: 0.1202 - dense_106_loss: 0.1726 - dense_107_loss: 0.2067 - dense_108_loss: 0.1229 - dense_109_loss: 0.1912 - dense_110_loss: 0.1346 - dense_111_loss: 0.3365 - dense_112_loss: 0.2075 - dense_113_loss: 0.2928 - dense_114_loss: 0.1149 - dense_115_loss: 0.1666 - dense_116_loss: 0.2663 - dense_117_loss: 0.1625 - dense_118_loss: 0.1654 - dense_119_loss: 0.6115 - dense_120_loss: 0.1790 - dense_121_loss: 0.0793 - dense_122_loss: 0.2200 - dense_123_loss: 0.1515 - dense_124_loss: 0.1853 - dense_105_acc: 0.9573 - dense_106_acc: 0.9492 - dense_107_acc: 0.9323 - dense_108_acc: 0.9629 - dense_109_acc: 0.9450 - dense_110_acc: 0.9604 - dense_111_acc: 0.8630 - dense_112_acc: 0.9302 - dense_113_acc: 0.8880 - dense_114_acc: 0.9715 - dense_115_acc: 0.9469 - dense_116_acc: 0.9135 - dense_117_acc: 0.9508 - dense_118_acc: 0.9506 - dense_119_acc: 0.6697 - dense_120_acc: 0.9490 - dense_121_acc: 0.9802 - dense_122_acc: 0.9252 - dense_123_acc: 0.9479 - dense_124_acc: 0.9446 - val_loss: 4.3609 - val_dense_105_loss: 0.1375 - val_dense_106_loss: 0.1974 - val_dense_107_loss: 0.1305 - val_dense_108_loss: 0.1605 - val_dense_109_loss: 0.1248 - val_dense_110_loss: 0.1403 - val_dense_111_loss: 0.4234 - val_dense_112_loss: 0.2145 - val_dense_113_loss: 0.3683 - val_dense_114_loss: 0.0958 - val_dense_115_loss: 0.1540 - val_dense_116_loss: 0.2918 - val_dense_117_loss: 0.2436 - val_dense_118_loss: 0.1595 - val_dense_119_loss: 0.5973 - val_dense_120_loss: 0.2372 - val_dense_121_loss: 0.0652 - val_dense_122_loss: 0.2132 - val_dense_123_loss: 0.1351 - val_dense_124_loss: 0.2710 - val_dense_105_acc: 0.9524 - val_dense_106_acc: 0.9464 - val_dense_107_acc: 0.9702 - val_dense_108_acc: 0.9464 - val_dense_109_acc: 0.9821 - val_dense_110_acc: 0.9762 - val_dense_111_acc: 0.8690 - val_dense_112_acc: 0.9405 - val_dense_113_acc: 0.8631 - val_dense_114_acc: 0.9821 - val_dense_115_acc: 0.9643 - val_dense_116_acc: 0.9048 - val_dense_117_acc: 0.9107 - val_dense_118_acc: 0.9524 - val_dense_119_acc: 0.6786 - val_dense_120_acc: 0.9286 - val_dense_121_acc: 0.9821 - val_dense_122_acc: 0.9286 - val_dense_123_acc: 0.9583 - val_dense_124_acc: 0.9107

Epoch 10/10

150/150 [==============================] - 31s 208ms/step - loss: 3.9869 - dense_105_loss: 0.1145 - dense_106_loss: 0.1668 - dense_107_loss: 0.2087 - dense_108_loss: 0.1206 - dense_109_loss: 0.1752 - dense_110_loss: 0.1339 - dense_111_loss: 0.3264 - dense_112_loss: 0.1992 - dense_113_loss: 0.2844 - dense_114_loss: 0.1201 - dense_115_loss: 0.1679 - dense_116_loss: 0.2541 - dense_117_loss: 0.1615 - dense_118_loss: 0.1555 - dense_119_loss: 0.5971 - dense_120_loss: 0.1945 - dense_121_loss: 0.0687 - dense_122_loss: 0.2138 - dense_123_loss: 0.1479 - dense_124_loss: 0.1760 - dense_105_acc: 0.9608 - dense_106_acc: 0.9479 - dense_107_acc: 0.9300 - dense_108_acc: 0.9625 - dense_109_acc: 0.9469 - dense_110_acc: 0.9621 - dense_111_acc: 0.8702 - dense_112_acc: 0.9338 - dense_113_acc: 0.8896 - dense_114_acc: 0.9684 - dense_115_acc: 0.9467 - dense_116_acc: 0.9161 - dense_117_acc: 0.9496 - dense_118_acc: 0.9502 - dense_119_acc: 0.6819 - dense_120_acc: 0.9435 - dense_121_acc: 0.9819 - dense_122_acc: 0.9260 - dense_123_acc: 0.9513 - dense_124_acc: 0.9454 - val_loss: 4.2460 - val_dense_105_loss: 0.1216 - val_dense_106_loss: 0.1987 - val_dense_107_loss: 0.1352 - val_dense_108_loss

```
s: 0.1300 - val_dense_109_loss: 0.1141 - val_dense_110_loss: 0.1368 - val_dense_111_loss: 0.3865 - val_dense_112_loss:
0.1870 - val_dense_113_loss: 0.3703 - val_dense_114_loss: 0.0810 - val_dense_115_loss: 0.1385 - val_dense_116_loss: 0.
2735 - val_dense_117_loss: 0.2119 - val_dense_118_loss: 0.1665 - val_dense_119_loss: 0.5980 - val_dense_120_loss: 0.24
76 - val_dense_121_loss: 0.0846 - val_dense_122_loss: 0.2319 - val_dense_123_loss: 0.1374 - val_dense_124_loss: 0.2951
- val_dense_105_acc: 0.9762 - val_dense_106_acc: 0.9405 - val_dense_107_acc: 0.9583 - val_dense_108_acc: 0.9643 - val_
dense_109_acc: 0.9762 - val_dense_110_acc: 0.9643 - val_dense_111_acc: 0.8512 - val_dense_112_acc: 0.9524 - val_dense_
113_acc: 0.8631 - val_dense_114_acc: 0.9881 - val_dense_115_acc: 0.9643 - val_dense_116_acc: 0.9048 - val_dense_117_ac
c: 0.9167 - val_dense_118_acc: 0.9405 - val_dense_119_acc: 0.7143 - val_dense_120_acc: 0.9345 - val_dense_121_acc: 0.9
643 - val_dense_122_acc: 0.9167 - val_dense_123_acc: 0.9583 - val_dense_124_acc: 0.9048
```

Out[402]: <keras.callbacks.History at 0x7f233a3d5b00>

In [403]:
```
test_generator.reset()
pred_model3=model.predict_generator(test_generator,
                                    steps=STEP_SIZE_TEST,
                                    verbose=1)
```

```
4952/4952 [==============================] - 73s 15ms/step
```

In [404]:
```
res3 = np.array(pred_model3)
pred_three = []
for m in range(len(test_generator)):
    prob = []
    for c in range(len(columns)):
        prob.append(res3[c][m][0])
    pred_three.append(prob)

pred_three = np.array(pred_three)

pred_bool = (pred_three >0.5)

predictions = pred_bool.astype(int)
#columns should be the same order of y_col
results = pd.DataFrame(predictions, columns=columns)
results["Filenames"] = test_generator.filenames
ordered_cols = ["Filenames"]+columns
results = results[ordered_cols]#To get the same column order
#results[results.bird==1]
results[:10]
```

Out[404]:

	Filenames	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	pottedplant	sheep	s
0	007157.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	002736.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	003624.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	002106.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	002927.jpg	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
5	008964.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	004589.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
7	001698.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	008567.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
9	005119.jpg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [405]:
```
#Saveing to harddrive
result = results
label = imageMap_test

prediction = result
actual = label

prediction = prediction.sort_values(by=['Filenames']).reset_index(drop=True)
actual = actual.sort_values(by=['Filenames']).reset_index(drop=True)

files = list(actual['Filenames'])
del prediction['Filenames']
del actual['Filenames']
pred_array = prediction.values
actual_array = actual.values
print('Total number of test samples: ',len(actual_array))

space = range(5,21)
for s in space:
    frac = int((s/20)*100)
    number = 0
    for n in range(len(actual_array)):
      if(((actual_array[n] == pred_array[n]).sum())>=s): number = number+1
    percent = number/len(actual_array)*100
    print('atleast ', frac,'% acurate predictions(',s,' out of 20) :',number,'  percentage: ',percent,'%')
```

```
Total number of test samples:  4952
atleast  25 % acurate predictions( 5  out of 20) : 4952   percentage:  100.0 %
atleast  30 % acurate predictions( 6  out of 20) : 4952   percentage:  100.0 %
atleast  35 % acurate predictions( 7  out of 20) : 4952   percentage:  100.0 %
atleast  40 % acurate predictions( 8  out of 20) : 4952   percentage:  100.0 %
atleast  45 % acurate predictions( 9  out of 20) : 4952   percentage:  100.0 %
atleast  50 % acurate predictions( 10  out of 20) : 4952   percentage:  100.0 %
atleast  55 % acurate predictions( 11  out of 20) : 4952   percentage:  100.0 %
atleast  60 % acurate predictions( 12  out of 20) : 4952   percentage:  100.0 %
atleast  65 % acurate predictions( 13  out of 20) : 4952   percentage:  100.0 %
atleast  70 % acurate predictions( 14  out of 20) : 4952   percentage:  100.0 %
atleast  75 % acurate predictions( 15  out of 20) : 4952   percentage:  100.0 %
atleast  80 % acurate predictions( 16  out of 20) : 4945   percentage:  99.85864297253634 %
atleast  85 % acurate predictions( 17  out of 20) : 4871   percentage:  98.36429725363489 %
atleast  90 % acurate predictions( 18  out of 20) : 4436   percentage:  89.5799676898223 %
atleast  95 % acurate predictions( 19  out of 20) : 2880   percentage:  58.15831987075929 %
atleast  100 % acurate predictions( 20  out of 20) : 556   percentage:  11.22778675282714 %
```

In [0]:
```
#Saveing to harddrive
results.to_csv('/content/drive/My Drive/Assignment6/Object Detection/result_model3.csv')
```