ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINERING DEPARTMENT

BLG 527E MACHINE LEARNING

CRN: 13817

Instructor: Zehra Çataltepe

Term Project

December 20, 2017

Tuğrul Yatağan

504161551

StudentName: Tuğrul Yatağan

StudentID: 504161551

Competition Name: Dog Breed Identification

kaggleID: tugrulyatagan

kaggleScore: 4.46075

kaggleScoreDate: December 20, 2017

methodNames: Keras CNN, OpenCV DNN

youtube videolink:

Running Code

dog.py file can be run with Python 2.7 interpreter. dog.py can be run directly without any command line arguments. tests.zip, train.zip, labels.csv.zip and saple_submission.csv.zip files must be extracted into same directory with dog,py

python dog.py

It took about 5 minutes to run. It is tested on Ubuntu 16.04 environment. Following Python modules needs to be installed to run dog.py;

pip install numpy

pip install pandas

pip install cv2

pip install sklearn

pip install opencv-contrib-python

pip install imutils

pip install tqdm

pip install keras

Dataset Description

Dataset contains images of 120 breeds of dogs. Training set has 10222 images and test set has 10357 images. Images has not only dogs but also surrounding environment and other objects like humans. Some images have multiple dogs with same breed in it. All images have different surrounding and different zoom and angle degree to dogs. All images have different resolution, brightness and dog scale. Most common and least common dog breeds in training set are:

Most commond breeds:

scottish_deerhound	126
maltese_dog	117
afghan_hound	116
entlebucher	115
bernese_mountain_dog	114

Least common breads:

golden_retriever	67
brabancon_griffon	67
komondor	67
briard	66
eskimo_dog	66

Dataset has similar dog breeds that classifying is challenging.

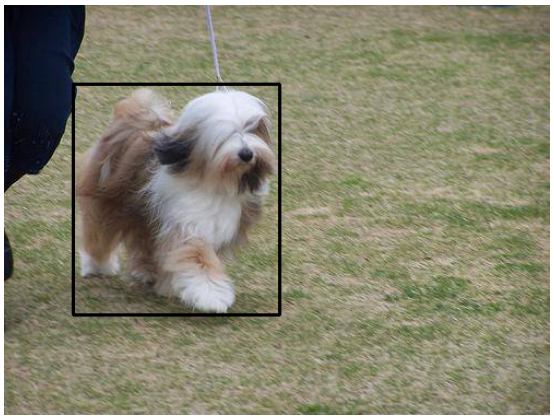
Brittany

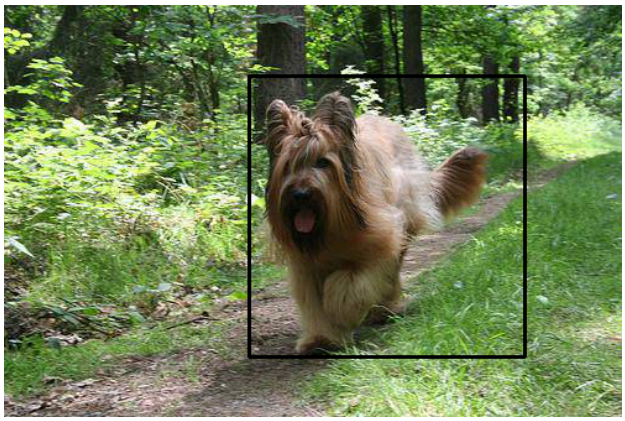
Welsh Springer Spaniel

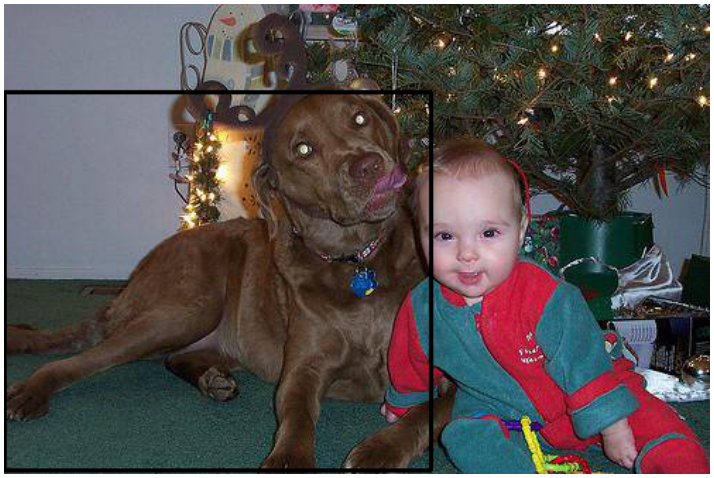
Methods Used

Dataset has images that have dogs in some portion of the image. So, I focused on detecting dog position in the image first because training on whole image can decrease classification accuracy since only a portion of the images has valuable data for dog breed classification. For this purpose I use OpenCV 3.3's deep neural network (dnn) library to detect dog positions on images. OpenCV dnn module has pretrained data(MobileNet SSD) to classify various objects like vehicles, humans, dogs and cats. This module use depthwise separable convolution to classify objects. The general idea behind depthwise separable convolution is to split convolution into a 3×3 depthwise convolution followed by a 1×1 pointwise convolution.

Example results of using OpenCV dnn MobileNet SSD to detect dogs:







To classify dog breeds, I use Keras CNN library on detected dog images from first step. CNN filters were halfed in each conv layer. There were max pooling layers between them. There were also dropout layers to prevent overfitting. Summary of the model:

Layer (type)	Output Shape	Param #	
conv2d_1 (Conv2D)	(None, 150, 150, 16)	208	INPUT
max_pooling2d_1 (MaxPooling2	(None, 75, 75, 16)	0	CONV
conv2d_2 (Conv2D)	(None, 75, 75, 32)	2080	POOL
max_pooling2d_2 (MaxPooling2	(None, 37, 37, 32)	0	CONV
conv2d_3 (Conv2D)	(None, 37, 37, 64)	8256	POOL
max_pooling2d_3 (MaxPooling2	(None, 18, 18, 64)	0	CONV
dropout_1 (Dropout)	(None, 18, 18, 64)	0	POOL
flatten_1 (Flatten)	(None, 20736)	0	DROPOUT
dense_1 (Dense)	(None, 500)	10368500	FLATTEN
dropout_2 (Dropout)	(None, 500)	0	DENSE
dense_2 (Dense)	(None, 120)	60120	DROPOUT
Total params: 10,439,164			DENSE
Trainable params: 10,439,164			OUTPUT
Non-trainable params: 0			

I use 150 x 150 as a input image size for CNN since larger than this requires too much RAM.

Experiment Results

Detection of dog positions on images with OpenCV dnn gives very accurate dog positions on images. There is no data to verify dog positions but I randomly choose 20 images and OpenCV dnn can detect dog positions on 19 images.

I split training dataset into 70%-30% for training and training validation set.

Training accuracy for CNN without dog detection is 10.2% which is very low. I got Kaggle score of 4.78474 with this method.

Training accuracy for CNN with dog detection preprocess is 13.7% which is not a quiet good achievement. I got Kaggle score of 4.46076 with this method. I assume (from Kaggle score) test accuracy for this method is a little better than first one.

Discussion

Result shows that dog breed classification with simple CNN method is not sufficient enough to get high classification accuracy. Dog detection before processing CNN has little effect on increasing CNN classification accuracy. Only simple CNN is not adequate to this kind of challenging image classification.

```
import numpy
import cv2
import imutils
import pandas
from tqdm import tqdm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model selection import train test split
from sklearn.metrics import accuracy score

import keras
from keras.models import Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Dropout, Flatten, Dense
from keras.models import Sequential


# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
           "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
           "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
           "sofa", "train", "tvmonitor"]

# load our serialized model from disk
net = cv2.dnn.readNetFromCaffe("MobileNetSSD deploy.prototxt.txt",
"MobileNetSSD deploy.caffemodel")

df train = pandas.read csv('labels.csv')
df test = pandas.read csv('sample submission.csv')

targets series = pandas.Series(df train['breed'])
one hot = pandas.get dummies(targets series, sparse = True)
one hot labels = numpy.asarray(one hot)

img size = 150


x train = []
y train = []
x test = []


for i, sample in enumerate(df train['id']):
    image = cv2.imread('train/' + sample + '.jpg')
    (h, w) = image.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 0.007843, (300,
    300), 127.5)

    # pass the blob through the network and obtain the detections and
    # predictions
    net.setInput(blob)
    detections = net.forward()

    # detect dog with highest confidence
    dogmatch = filter(lambda x: x[1] == 12, detections[0][0])
    # if dog is not detected try similar classes
    if len(dogmatch) == 0:
        dogmatch = filter(lambda x: x[1] in (8, 3, 17, 13, 10), detections[0][0])

    if len(dogmatch) > 0:
        box = dogmatch[0][3:7] * numpy.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int").clip(min=0)
        image = image[startY:endY, startX:endX]

    x train.append(cv2.resize(image, (img size, img size)))
    y train.append(one hot labels[i])


for i, sample in enumerate(df test['id']):
    image = cv2.imread('test/' + sample + '.jpg')
    (h, w) = image.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 0.007843, (300,
    300), 127.5)
```

```
72
73      # pass the blob through the network and obtain the detections and
74      # predictions
75      net.setInput(blob)
76      detections = net.forward()
77
78      # detect dog with highest confidence
79      dogmatch = filter(lambda x: x[1] == 12, detections[0][0])
80      # if dog is not detected try similar classes
81      if len(dogmatch) == 0:
82          dogmatch = filter(lambda x: x[1] in (8, 3, 17, 13, 10), detections[0][0])
83
84      if len(dogmatch) > 0:
85          box = dogmatch[0][3:7] * numpy.array([w, h, w, h])
86          (startX, startY, endX, endY) = box.astype("int").clip(min=0)
87          image = image[startY:endY, startX:endX]
88
89      x test.append(cv2.resize(image, (img size, img size)))
90
91
92  y train raw = numpy.array(y train, numpy.uint8)
93  x train raw = numpy.array(x train, numpy.float32) / 255.
94  x test  = numpy.array(x test, numpy.float32) / 255.
95
96  X train, X test, y train, y test = train test split(x train raw, y train raw,
    test size=0.3, random state=1)
97
98  model = Sequential()
99  model.add(Conv2D(filters=16, kernel size=2, padding='same', activation='relu',
100                          input shape=(img size, img size, 3)))
101 model.add(MaxPooling2D(pool size=2))
102 model.add(Conv2D(filters=32, kernel size=2, padding='same', activation='relu'))
103 model.add(MaxPooling2D(pool size=2))
104 model.add(Conv2D(filters=64, kernel size=2, padding='same', activation='relu'))
105 model.add(MaxPooling2D(pool size=2))
106 model.add(Dropout(0.3))
107 model.add(Flatten())
108 model.add(Dense(500, activation='relu'))
109 model.add(Dropout(0.4))
110 model.add(Dense(120, activation='softmax'))
111
112 model.compile(optimizer='rmsprop', loss='categorical crossentropy',
    metrics=['accuracy'])
113
114
115 model.summary()
116
117
118 model.fit(X train, y train, epochs=1, validation data=(X test, y test), verbose=1)
119
120 preds = model.predict(x test, verbose=1)
121
122 sub = pandas.DataFrame(preds)
123 # Set column names to those generated by the one-hot encoding earlier
124 col names = one hot.columns.values
125 sub.columns = col names
126 # Insert the column id from the sample submission at the start of the data frame
127 sub.insert(0, 'id', df test['id'])
128 sub.to csv('out.csv', index=False)
129
```