Второй отборочный этап

Задача 1. Распознавание сигналов светофора (30 баллов)

В этом задании необходимо использовать свои знания по компьютерному зрению для создания классификатора изображений светофора! Даны изображения светофора, на каждом из которых горит только один из трех сигналов: красный, желтый или зеленый.

Система классификации

Подготовлена программа на языке Python, в которой участники предварительно обрабатывают изображения, выделяют особенности, которые помогают находить отличия в видах изображения, используют эти особенности, чтобы разделить изображения по трем категориям: светофор с красным, желтым или зеленым сигналом.

Этапы работы:

0.1. Посмотреть видео-курс http://newgen.education/video/trafficlights.
0.2. Скачать программу https://yadi.sk/d/C3LmnDnF4u2pwA.
0.3. Посмотреть короткое видео по работе с площадкой https://www.youtube.com/watch?v=9ZD52fzCulA.

Загрузка и визуализация данных

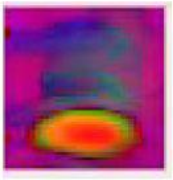
В любой задаче по классификации сначала необходимо ознакомиться с данными: необходимо загрузить изображения сигналов светофоров и визуализировать их!

Предварительная обработка

Входные изображения и выходные метки (labels) должны быть стандартизированы: все входные данные должны быть одного типа и одного размера, а выходные данные должны быть числовой меткой. Таким образом можно проанализировать все входные изображения одним и тем же способом и предугадать, чего следует ожидать для нового изображения.

Выделение особенностей

Теперь необходимо выделить особенности в каждом изображении и классифицировать их. Поле для творчества безгранично: объекты могут быть как одномерными массивами (векторами), так и отдельными значениями, которые дают некую информацию об изображении, чтобы помочь вам классифицировать его как красный, желтый или зеленый сигнал светофора.

 ------->	 ------->	
Загруженное изображение	HSV	V - яркость

Ошибки классификации и визуализации

Наконец, создана функция, которая использует найденные особенности для классификации любого сигнала светофора. Задача функции будет заключаться в том, чтобы получать на вход изображение и выводить метку (вектор из 3 значений). Необходимо сравнить прогнозируемую метку с истинной меткой и определить точность классифицируемой модели.

Оцените свою модель

Чтобы работа была засчитана, классификатор должен иметь точность $> 70\%$. Вероятнее всего, нужно будет повысить точность классификатора путем изменения имеющихся особенностей изображения или добавления новых. Чем выше точность, тем больше баллов получит команда за решение.

Решение

Сформулируем задачу распознавания сигналов светофора. Мы имеем ряд изображений светофоров, на которых включен тот или иной сигнал. Нам следует определить, какой сигнал включен. Уточним, что это не изображение улицы, на котором где-то есть светофор — это светофор уже вырезанный из какого-то более общего кадра.

Следует отметить, что найти на изображении светофор и вырезать из этого изображения прямоугольную область, в которую этот светофор вписан, является самостоятельной задачей. Такая задача называется детектированием и предшествует распознаванию.

То есть, решая задачу распознавания, мы работаем с заранее детектированными и вырезанными изображениями светофоров. Как они были получены — отдельная тема, которую мы рассмотрим позже.

Итак, определяем какой сигнал включен на изображениях светофоров.

Самое интересное в нашем алгоритме то, что определяя цвет сигнала светофора, параметром «цвет» мы даже и не пользуемся. Зная, в какой области какой цвет расположен на светофоре, мы по яркости этой области определяем цвет. Причем на черно-белом изображении. Алгоритм такой:

- приводим все изображения со светофорами к одному размеру;
- на каждом изображении отрезаем фон по краям, иногда даже с частью светофора;

- обрезанное изображение переводим в формат HSV;
- делаем HSV изображение одноканальным, оставив из трех каналов один — яркость;
- считаем суммы значений яркости для трех областей изображения: «красной», «желтой» и «зеленой»;
- по максимальной сумме определяем в какой области включен сигнал.

Рассмотрим исходный текст программы.

```
import cv2                  #импорт библиотеки OpenCV
import numpy                  #импорт библиотеки NumPy (работа с массивами)
for i in range(1,6):
    #цикл по количеству изображений, у нас 5
    Frame = cv2.imread(str(i) + ".jpg")
    #считываем изображение из файла, у которого имя совпадает с номером,
    #в переменную Frame
    Frame = cv2.resize(Frame, (40,100))
    #меняем размер изображения
    cv2.imshow("Traffic Lights " + str(i), Frame)
    #выводим его в окно ''TrafficLight"
    Cuted_Frame = Frame[4:95, 4:35]
    #обрезаем края изображения
    cv2.imshow("Cuted Frame " + str(i), Cuted_Frame)
    #выводим в окно "CutedFrame"
    HSV = cv2.cvtColor(Cuted_Frame, cv2.COLOR_BGR2HSV)
    #переводим в HSV
    Brightness_Only = HSV[:, :, 2]
    #оставляем канал яркости
    cv2.imshow("Brightness Only " + str(i), Brightness_Only)
    #выводим в окно "Brightness Only "
    Red_Sum_Brightness = numpy.sum(Brightness_Only[0:30, 0:30])
    Yellow_Sum_Brightness = numpy.sum(Brightness_Only[31:60, 0:30])
    Green_Sum_Brightness = numpy.sum(Brightness_Only[61:90, 0:30])
    #суммируем яркости в областях каждого цвета
    cv2.rectangle(Cuted_Frame, (0, 0), (30, 30), (0, 0, 255), 1)
    cv2.rectangle(Cuted_Frame, (0, 31), (30, 60), (0, 255, 255), 1)
    cv2.rectangle(Cuted_Frame, (0, 61), (30, 90), (0, 255, 0), 1)
    #для наглядности обводим каждую область линией своего цвета
    cv2.imshow("Areas " + str(i),Cuted_Frame)
    #и выводим в окно "Areas"
    print(str(Red_Sum_Brightness) + " : " + str(Yellow_Sum_Brightness) + " : " +
       str(Green_Sum_Brightness))
    #выводим на печать суммы яркости для всех трех областей
    if Red_Sum_Brightness>Yellow_Sum_Brightness:
        if Red_Sum_Brightness>Green_Sum_Brightness:
            print('Red')
        else:
            print('Green')
    elif Green_Sum_Brightness>Yellow_Sum_Brightness:
        print('Green')
    else:
        print('Yellow')
    #определяем для области какого цвета сумма максимальна и выводим на печать цвет
    cv2.moveWindow("Traffic Lights "+str(i),50,50)
    cv2.moveWindow("Cuted Frame "+str(i),350,150)
    cv2.moveWindow("Brightness Only "+ str(i),650,250)
    cv2.moveWindow("Areas "+str(i),950,350)
    #все окна располагаем удобным образом
    if cv2.waitKey() == 27:
```

```
51          break
52      #нажата кнопка "Esc"?
53      #тогда выход из цикла,
54      #нет --- следующий кадр
55  cv2.destroyAllWindows()
56  #закрываем все окна
```

Запустив программу, можно убедиться в том, что сумма яркости пикселей области с включенным сигналом значительно больше сумм двух других областей, и алгоритм работает без ошибок.

Неудобство этой программы в том, что она работает с файлами изображений со специфическими именами. В реальной жизни приходится иметь дело с файлами, имеющими какие угодно имена, причем заранее, их знать, мы не можем и не должны.

Задача 2. Распознавание дорожных знаков (30 баллов)

Создание классификатора изображений дорожных знаков.

Дано

Набор изображений дорожных знаков, состоящий из 8 классов, а также класс, не содержащий изображений дорожных знаков.

Задание

Обработать изображения, извлечь параметры, которые помогут различить разные типы знаков и использовать эти параметры для классификации изображений на 8 классов, а также убрать из классификации изображения, не содержащие дорожные знаки.

Выполнение

В папке с заданием находятся вспомогательные файлы и инструкции. Для каждой функции написаны комментарии, позволяющие определить ее назначение, входные параметры и форматы выходных данных, являющиеся результатом работы функции. Внимательно ознакомьтесь с комментариями к функциям!

Скачать папку с заданием https://yadi.sk/d/UBL67SbQ9_ybog.

Для работы с проектом необходимо установить python 3 версии, opencv 3 версии, а также библиотеку numpy. Все функции для распознавания представлены в файле eval.py. Их можно изменять. Названия функций и самого файла при этом должны оставаться неизменными. Выборка, с которой работает алгоритм, изменяется в файле main.py: функция load_data и соответственно массивы данных, получаемые из этой функции. Файл helpers.py изменению не подлежит!

Обучающий видео-курс http://newgen.education/video/roadsigns.

Проверка кода осуществляется на тестовых данных: изображениях, не включенных в тренировочную и валидационную выборки. Основной метрикой оценки явля-

ется точность работы классификатора. При одинаковой точности оценивается также скорость работы классификатора.

Решение

Задача может быть решена простым сравнением с эталоном. Получение эталона для изображения знака:

```
no_drive = cv2.imread("data/standards/no_drive.png")
no_drive = cv2.resize(no_drive, (64, 64))
#no_drive = cv2.inRange(no_drive, (0, 89, 155), (255, 255, 255))
no_drive = cv2.GaussianBlur(no_drive, (5, 5), 0)
no_drive = cv2.cvtColor(no_drive, cv2.COLOR_BGR2GRAY)
ret2, no_drive = cv2.threshold(no_drive, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
#no_drive = cv2.erode(no_drive, None, iterations=2)
```

К изображению знака, которое необходимо классифицировать применяем те же операции и сохраняем в переменную sign. Далее попиксельно сравниваем его с эталоном:

```
no_drive_val = 0
for i in range(64):
    for j in range(64):
        if sign[i][j] == no_drive[i][j]:
            no_drive_val += 1
```

Если совпадений с эталоном больше определенного процента, то считаем что знак распознан.

```
if res == no_drive_val and res > 2700:
    predicted_label = "no_drive"
```

Для решения задачи, придется сравнить знак с восьмью эталонами. И посчитать процент совпадения с каким больше всего.

Задача 3. (30 баллов)

Создание детектора пешеходов на изображении.

Дано

Набор изображений, на которых присутствуют пешеходы.

Задание

Обработать изображения, извлечь параметры, которые помогут выделить общие признаки пешеходов, и использовать эти признаки для детектирования пешеходов на изображениях.

Выполнение

В папке с заданием находятся вспомогательные файлы и инструкции. Для каждой функции написаны комментарии, позволяющие определить ее назначение, входные параметры и форматы выходных данных, являющиеся результатом работы функции. Внимательно ознакомьтесь с комментариями к функциям!

Скачать папку с заданием https://yadi.sk/d/Ia6BTV9rpuaNwQ.

Для работы с проектом необходимо установить Python версии 3+, OpenCV версии 3+, а также библиотеку numpy.

Все функции для распознавания представлены в файле eval.py. Их можно изменять. Названия функций и самого файла при этом должны оставаться неизменными. Если вы используете нейросетевые или другие модели, загрузите их в файле eval.py в специально обозначенном месте.

Выборка, с которой работает алгоритм, изменяется в файле main.py: функция load_data и массивы данных, возвращаемые из этой функции. Файл helpers.py изменению не подлежит!

Обучающий видео-курс https://avt.global/cv#pedestrians.

Проверка кода осуществляется на тестовых данных: изображениях, не включенных в тренировочную и валидационную выборки. Основной метрикой оценки является точность работы классификатора и ложные срабатывания. Оценка корректного детектирования производится методом Intersection over Union (IoU). Финальная точность вычисляется по формуле $(Tp/n - Fp/n)$, где: Tp - количество правильных решений детектора ($Accuracy$), Fn — количество ложных срабатываний, n — общее количество объектов При одинаковой точности оценивается также скорость работы классификатора.

Решение

Детектировать пешеходов можно с помощью SVM- детектора, который предварительно нужно обучить. Для этого участникам предоставлены размеченные изображения с пешеходами. Обучение детектора выполняется при помощи следующе кода:

```
import dlib
import os
import cv2
import xml.etree.ElementTree as pars


#адрес к датасету
dir=r"C:\Users\DFCZ\PycharmProjects\DetektionWithLabelImage\AllPeople"
images=[]
annots=[]

ImgNameList = os.listdir(dir + "\images")
print (ImgNameList)


# перебираем изображения по одному, каждому изображению ставим в соответствие аннотации (координаты пешехода на изображении)
for FileName in ImgNameList:
    image=cv2.imread(dir+"/images/"+FileName)
```

```
    image=cv2.cvtColor(image,cv2.COLOR_BGR2RGB)

    OnlyFileName=FileName.split(".")[0]
    print (OnlyFileName)
    e = pars.parse(dir+"/annotations/xmls/"+OnlyFileName+".xml")
    root=e.getroot()

    #object=root.find("object")
    for object in root.findall("object"):
        object=object.find("bndbox")

        x=int(object.find("xmin").text)
        y = int(object.find("ymin").text)
        x2 = int(object.find("xmax").text)
        y2 = int(object.find("ymax").text)

        if (x2 - x) / (y2 - y) < 0.7:
            images.append(image)
            annots.append([dlib.rectangle(left=x, top=y, right=x2, bottom=y2)])


# сформированные масивы с изображениями и аннотациями подаем на вход функции обучения
↪  детектора. Обученный детектор сохраняем в файл

options = dlib.simple_object_detector_training_options()
options.be_verbose=True
detector = dlib.train_simple_object_detector(images, annots, options)

detector.save("tld.svm")
print ("Detector Saved")
```

После того как детектор обучен, код для его применения следует встроить в файл eval.py. Пример использования обученного детектора:

```
import cv2
import dlib

model_detector = dlib.simple_object_detector("tld.svm")

cam=cv2.VideoCapture(0)

while (1):
    ret,frame=cam.read()

    boxes = model_detector(frame)
    for box in boxes:
        print (box)
        (x, y, xb, yb) = [box.left(), box.top(), box.right(), box.bottom()]
        cv2.rectangle(frame, (x, y), (xb, yb), (0, 0, 255), 2)

    cv2.imshow("Frame",frame)

    key = cv2.waitKey(1)
    if cv2.waitKey(1)==ord('q'):
        break

cv2.destroyAllWindows()
cam.release()
```

Задача 4. Распознавание дорожных знаков (30 баллов)

Создание классфикатора цифр.

Дано

Массив изображений с цифрами.

Задание

Определить цифру на изображении.

Выполнение

Скачайте задание: https://yadi.sk/d/C105blmAcFJ9OQ. В архиве находятся:

- программа на языке Python для обработки изображений и настройки собственного алгоритма классификации (файлы main.py, eval.py),
- папку pictures_numeric_train с тренировочным набором данных,
- папку pictures_numeric_val с валидационным набором данных.

В файле eval.py следует заполнить 3 функции:

1. Функцию предобработки данных.
2. Фунцию загрузки модели.
3. Функцию предсказания цифр на картинке: вектор цифр в формате списка, например [1, 2], если ваш алгоритм предсказывает цифры 1 и 2 на конкретной картинке. **На картинке может быть от 1 до 3 цифр**.

Не меняйте названия функций файла eval.py.

В файле main.py вы сможете проверить работоспособность модели и ее точность на валидационной выборке. Если модель работает и ее точность вас устраивает, то вы можете отправить решенное задание.

Проверка кода осуществляется на тестовых данных: изображениях, не включенных в тренировочную и валидационную выборки. **Точность модели рассчитывается следующим образом**:

Сумма количества цифр правильно определенных векторов к общему числу цифр на всех векторах. Например, правильный вектор ответов [[1], [1, 2], [1, 2, 3], [3, 2, 1]], а предсказанный [[1], [1, 3], [1, 3, 2], [3, 2, 1]]. Правильно определены 1-ый и 4-ый вектора, количество их цифр составляет 4, а общее количество цифр — 9. Точность составляет 4 / 9, то есть 0,4444...

Для того, чтобы ваша модель была засчитана, вы должны будете получить точность классификатора больше 0,5.

В качестве решения присылайте архив с файлом eval.py и с сохраненной вами моделью в нужном формате. Если вы планируете использовать библиотеку tensorflow, то модель нужно выгружать в формате .h5 (Как сохранять модель в формат h5 https://www.tensorflow.org/tutorials/keras/save_and_load?hl=ru). **Архив должен называться num_task.zip**.

Важно:

1. Для построения модели нейронной сети используйте tensorflow версии 2.0.1.
2. Версия sklearn на сервере 0.19.1, а dlib 19.19.0. Для импорта этих моделей используйте pickle.
3. Проверьте, что в векторе с предсказанием данные в формате int, а не float или str, то есть вектор должен быть в формате [1, 2], а не [1.0, 2.0] или ['1', '2'].
4. Архив с заданием должен называться num_task.zip и содержать файлы eval.py и вашу модель

Решение

Для решения задачи необходимо детектировать цифры на изображении и распознать их. Детектировать цифры можно по их цвету: все цифры черного цвета, изображены на белом фоне. Однако, на изображениях есть объекты черного цвета, помимо цифр. Поэтому, сначала детектируем белые кубики, а уже на них детектируем черные цифры.

Детектирование по цвету выполняется в три этапа: бинаризация, поиск контуров, выбор контуров, подходящих по размерам и форме. Все эти этапы представлены в следующем коде:

```
image = cv.read("image.jpg")
blur = cv.blur(image, (5, 5))
#бинаризация изображения
thresh = cv.inRange(blur, (190, 190, 190), (255, 255, 255)) # Пороги бинаризации
thresh = cv.erode(thresh, None, iterations=2)
thresh = cv.dilate(thresh, None, iterations=5)

#поиск контуров
contours = cv.findContours(thresh, cv.RETR_TREE, cv.CHAIN_APPROX_NONE)
contours = contours[0]    # 0 или 1 в разных версиях OpenCV

#перебор контуров
if contours:
    contours = sorted(contours, key=cv.contourArea, reverse=True)
    for cont in contours:
        # отбор контуров по площади
        cont_area = cv.contourArea(cont)
        if cont_area > 1000 and cont_area< 11000:

        rect = cv.minAreaRect(contours[0])
        box = np.int0(cv.boxPoints(rect))
        (x, y, w, h) = cv.boundingRect(contours[0])

        # вырезание интересующего нас объекта на изображении
        roiImg = frameBase[y:y + h, x:x + w]
```

После того, как цифры детектированы, следует провести распознавание и понять, какая именно цифра перед нами. Цифры достаточно простой объект и его можно распознать сравнением с эталоном. Поскольку цифра может быть повернута одним из четырех способов, для каждой цифры необходимо 4 эталонных изображения. Эталоны можно подготовить из предоставленного набора изображений. Следующий код демонстрирует распознавание сравнением с эталоном:

```
one_val = 0
two_ val = 0

#для изображения 64 на 64 пикселя
for i in range(64):
    for j in range(64):
        if (roiImg[i][j]==one_template[i][j]):
            one_val += 1
        if (roiImg [i][j]== two_template [i][j]):
            two_ val += 1

print (one_val)
print (two_val)
```

Итоговое решение о том, сколько цифр и в каком порядке они расположены, вы принимаете по результатам распознавания. Число распознанных объектов — число цифр. Порядок цифр определяется сравнением X-координат детектированных объектов. Чем меньше X, тем правее располагается цифра.