Семантическая сегментация изображений на основе метода машинного обучения

Подготовил: Синюков Максим Юрьевич
Руководитель: Клименко Сергей Владимирович

Предмет исследования

Алгоритмы определяющие набор пикселей на изображении к определенным классам

```
def circle_points(resolution, center,
radius):

    radians = np.linspace(0, 2*np.pi,
resolution)
c = center[1] +
radius*np.cos(radians)#polar co-ordinates
    r = center[0] + radius*np.sin(radians)

    return np.array([c, r]).T

  points = circle_points(200, [80, 250],
  80)[:-1]

fig, ax = image_show(image)
  ax.plot(points[:, 0], points[:, 1],
  '--r', lw=3)
```

Пример алгоритма на основе активной контурной сегментации

Цели и задачи исследования:

Цель: Проведение семантической сегментации изображения и обучение модели классификации.

Задачи:

1) Найти и исследовать существующие методы сегментации изображений
2) Создать модель классификации для сегментации входных изображений.

Что такое машинное обучение?

Машинное обучение — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач.

Задачи решаемые машинным обучением

1)Регрессия

2)Классификация

3)Кластеризация

4)Уменьшение размерности

5)Выявление аномалий

6)Прогнозирование

Типы машинного обучения

-С учителем

-Без учителя

-С подкреплением

Пример работы алгоритма классификации

```
# Вспомогательная функция
def decode_segmap(image, source, bgimg, nc=21):

  label_colors = np.array([(0, 0, 0),  # 0=background
               # 1=aeroplane, 2=bicycle, 3=bird, 4=boat, 5=bottle
               (128, 0, 0), (0, 128, 0), (128, 128, 0), (0, 0, 128), (128, 0, 128),
               # 6=bus, 7=car, 8=cat, 9=chair, 10=cow
               (0, 128, 128), (128, 128, 128), (64, 0, 0), (192, 0, 0), (64, 128, 0),
               # 11=dining table, 12=dog, 13=horse, 14=motorbike, 15=person
               (192, 128, 0), (64, 0, 128), (192, 0, 128), (64, 128, 128), (192, 128, 128),
               # 16=potted plant, 17=sheep, 18=sofa, 19=train, 20=tv/monitor
               (0, 64, 0), (128, 64, 0), (0, 192, 0), (128, 192, 0), (0, 64, 128)])

  r = np.zeros_like(image).astype(np.uint8)
  g = np.zeros_like(image).astype(np.uint8)
  b = np.zeros_like(image).astype(np.uint8)

  for l in range(0, nc):
    idx = image == l
    r[idx] = label_colors[l, 0]
    g[idx] = label_colors[l, 1]
    b[idx] = label_colors[l, 2]


  rgb = np.stack([r, g, b], axis=2)
```

Работа с изображением для дальнейшей работы с ним

```
# Загружаем изображения для сегментации
foreground = cv2.imread(source)

# загружаем изображение заднего плана
background = cv2.imread(bgimg)

# Изменение изображения на RGB и изменение его размера в соответствии с формой R-диапазона на выходной карте RGB
foreground = cv2.cvtColor(foreground, cv2.COLOR_BGR2RGB)
background = cv2.cvtColor(background, cv2.COLOR_BGR2RGB)
foreground = cv2.resize(foreground,(r.shape[1],r.shape[0]))
background = cv2.resize(background,(r.shape[1],r.shape[0]))


# Конвертируем uint8 во float
foreground = foreground.astype(float)
background = background.astype(float)

# Создаем бинарную маску выходной карты RGB используя значение 0
th, alpha = cv2.threshold(np.array(rgb),0,255, cv2.THRESH_BINARY)

# нанесение на края маски размытия по Гауссу для смегчения контура
alpha = cv2.GaussianBlur(alpha, (7,7),0)

# Нормализация альфа маски
alpha = alpha.astype(float)/255

# Используем alpha matte на сегментированное изображение
foreground = cv2.multiply(alpha, foreground)

# умножение фона на 1-alpha
background = cv2.multiply(1.0 - alpha, background)

# Добавление маски на передний и задний фон
outImage = cv2.add(foreground, background)
```

Вывод результата работы

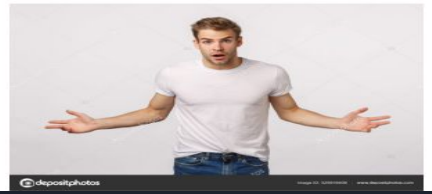
```
  trf = T.Compose([T.Resize(400),
                   T.ToTensor(),
                   T.Normalize(mean = [0.485, 0.456, 0.406],
                               std = [0.229, 0.224, 0.225])])
  inp = trf(img).unsqueeze(0).to(dev)
  out = net.to(dev)(inp)['out']
  om = torch.argmax(out.squeeze(), dim=0).detach().cpu().numpy()

  rgb = decode_segmap(om, path, bgimagepath)

  plt.imshow(rgb); plt.axis('off'); plt.show()


dlab = models.segmentation.deeplabv3_resnet101(pretrained=1).eval()

segment(dlab, './1.png','./12.png', show_orig=False)
segment(dlab, './1.png','./12.png', show_orig=True)
```



Результаты исследования

- Понимание как происходит сегментация изображений
- Применение полученных знаний на практике путем создания системы семантической сегментации изображений

Спасибо за внимание