

Компьютерное зрение

Практический курс
Савельева Юлия Олеговна
i.o.saveleva.kpfu@gmail.com
2-й семестр, 24(29).04.2021 г.



Class Activation Maps

Idea

Последней задачей курса будет Weakly-supervised object localization на основе Class Activation Maps (CAM). В статье, приведенной на следующем слайде, был предложен метод, с помощью которого мы можем построить heatmap для активаций нейронной сети и таким образом увидеть, какая именно область изображения повлияла на итоговое значение выхода нейронной сети для конкретного класса. В следующем задании этот heatmap будет использоваться для выделения приблизительных границ объекта, с последующим подсчетом точности локализации с помощью mean Average Precision.

Так как ResNet предобучен на ImageNet, то нам необходимо дообучить его на датасете, у которого есть gt bounding boxes, для простоты возьмем уже использованный нами COCO (пример датасета для классификации на COCO приведен на следующем слайде).

Class Activation Maps

Sources

Пример COCO Classification Dataset:

https://github.com/mingming97/coco-classification/blob/master/coco_80_dataset.py

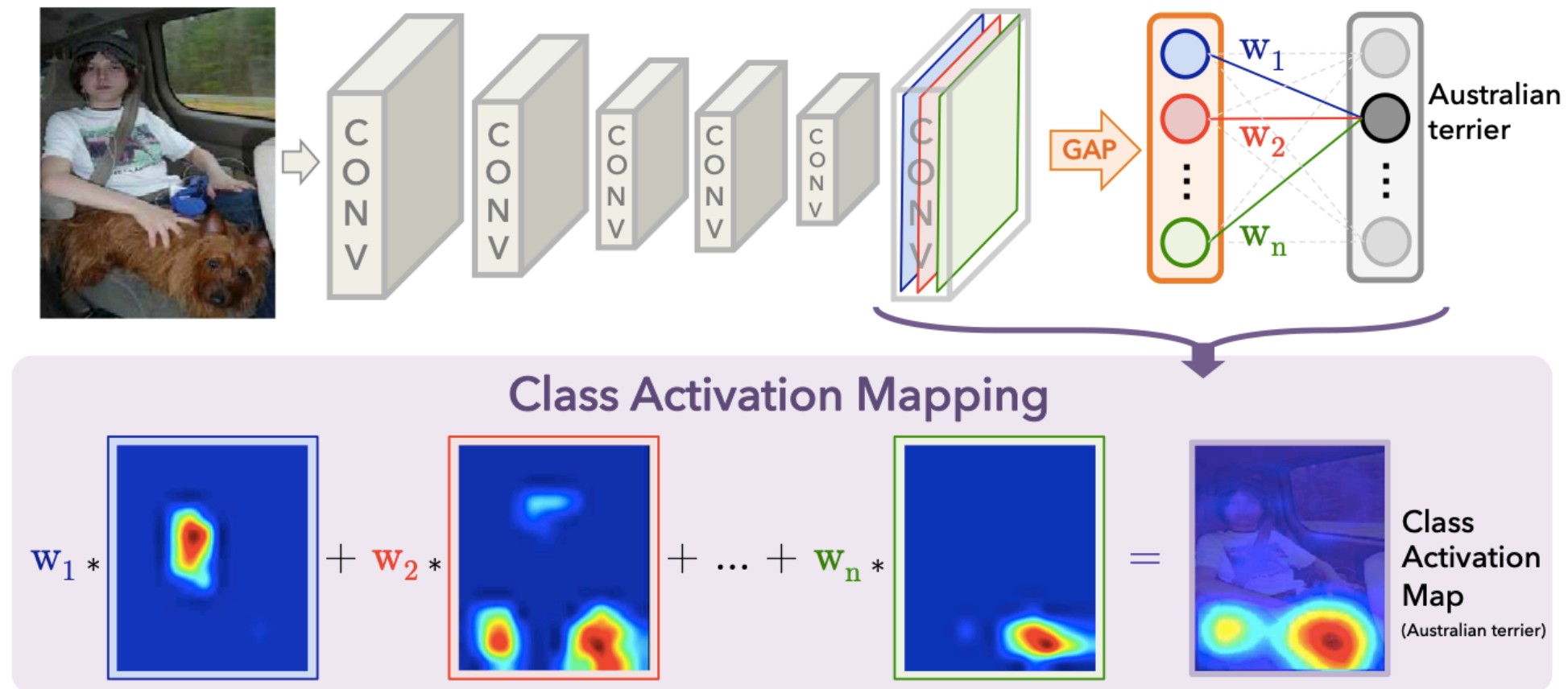
Базовая статья по CAM:

http://cnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf

- 1) Дообучить ResNet-50 от Pytorch предобученный на ILSVRC на классификацию COCO
- 2) Реализовать CAM для дообученной сети, приложить к проекту результаты работы для нескольких картинок

Class Activation Maps

Scheme



Class Activation Maps

Code

Чтобы получить CAM:

input_tensor - (B=1, C=3, H, W)

```
target_layer = model.layer4[-1] # layer before GAP in ResNet50
fc = model.fc.weights.cpu().data.numpy()
self.activations_and_grads = ActivationsAndGradients(model,
target_layer)
output = self.activations_and_grads(input_tensor)
target_category = np.argmax(output.cpu().data.numpy()) # could
be any class, we just take argmax for the most confident output
activations =
self.activations_and_grads.activations[-1].cpu().data.numpy()[0, :]
weights = fc[target_category]
cam = np.zeros(activations.shape[1:])
for i, w in enumerate(weights):
    cam += w * activations[i, :, :]
```

Чтобы получить активации слоя:

```
class ActivationsAndGradients:
    def __init__(self, model, target_layer):
        self.model = model
        self.activations = []
        target_layer.register_forward_hook(self.save_activation)

    def save_activation(self, module, input, output):
        activation = output
        if self.reshape_transform is not None:
            activation = self.reshape_transform(activation)
        self.activations.append(activation.cpu())

    def __call__(self, x):
        self.activations = []
        return self.model(x)
```

Class Activation Maps

Visualization

Чтобы получить итоговый heatmap:

```
cam = np.maximum(cam, 0)
cam = cv2.resize(cam, (img_h, img_w))
cam = cam - np.min(cam)
cam = cam / np.max(cam)
result = show_cam_on_image(img, cam)
```



Визуализация heatmap на картинке:

```
def show_cam_on_image(img, cam):
    heatmap =
cv2.applyColorMap(np.uint8(255 *
mask), cv2.COLORMAP_JET)
    heatmap = np.float32(heatmap) / 255
    cam = heatmap + img
    cam = cam / np.max(cam)
    return np.uint8(255 * cam)
```

Дедлайн 12.05.2020 23:59