МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального

приборостроения

Отчет по лабораторной работе №3

по дисциплине: «Анализ защищенности систем искусственного интеллекта»

Выполнил:
Студент группы ББМО-01-22
ФИО: Карев Д.П.

Москва 2023

1. Установим tf-keras-vis.

```
[1] !pip install tf-keras-vis
```

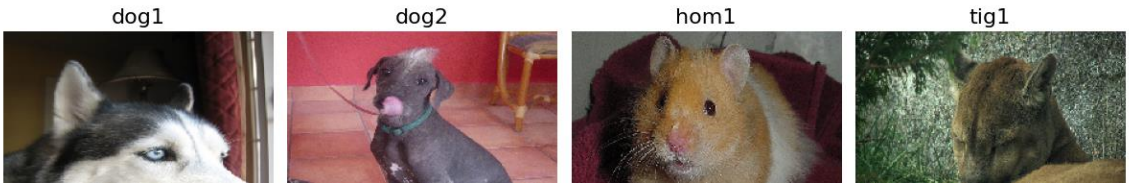
```
Collecting tf-keras-vis
  Downloading tf_keras_vis-0.8.6-py3-none-any.whl (52 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.1/52.1 kB 1.7 MB/s eta 0:00:00
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis) (1.11.4)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis) (9.4.0)
Collecting deprecated (from tf-keras-vis)
  Downloading Deprecated-1.2.14-py2.py3-none-any.whl (9.6 kB)
Requirement already satisfied: imageio in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis) (2.31.6)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis) (23.2)
Requirement already satisfied: wrapt<2,>=1.10 in /usr/local/lib/python3.10/dist-packages (from deprecated->tf-keras-vis) (1.14.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from imageio->tf-keras-vis) (1.23.5)
Installing collected packages: deprecated, tf-keras-vis
Successfully installed deprecated-1.2.14 tf-keras-vis-0.8.6
```

```
[2] %reload_ext autoreload
    %autoreload 2

    import numpy as np
    from matplotlib import pyplot as plt
    %matplotlib inline

    import tensorflow as tf
    from tf_keras_vis.utils import num_of_gpus

    _, gpus = num_of_gpus()
    print('Tensorflow recognized {} GPUs'.format(gpus))
```

```
Tensorflow recognized 0 GPUs
```

2. Загрузим 4 изображения из датасета ImageNet и отобразим их.

```
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.applications.vgg16 import preprocess_input


image_titles = ['dog1', 'dog2', 'hom1', 'tig1']


img1 = load_img('dog1.jpeg', target_size=(224, 224))
img2 = load_img('dog2.jpeg', target_size=(224, 224))
img3 = load_img('hom1.jpeg', target_size=(224, 224))
img4 = load_img('tig1.jpeg', target_size=(224, 224))
images = np.asarray([np.array(img1), np.array(img2), np.array(img3), np.array(img4)])

X = preprocess_input(images)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))
for i, title in enumerate(image_titles):
    ax[i].set_title(title, fontsize=16)
    ax[i].imshow(images[i])
    ax[i].axis('off')
plt.tight_layout()
plt.show()
```



3. Сделаем класс изображения, для примера сделаем все 99.

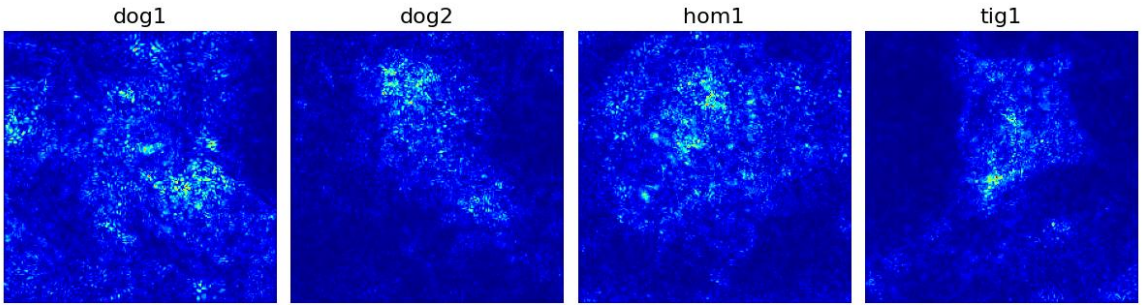
```
[6] from tf_keras_vis.utils.model_modifiers import ReplaceToLinear
    replace2linear = ReplaceToLinear()
    def model_modifier_function(cloned_model):
        cloned_model.layers[-1].activation = tf.keras.activations.linear
```

```
[8] from tf_keras_vis.utils.scores import CategoricalScore
    score = CategoricalScore([99, 99, 99, 99])
```

0 сек.

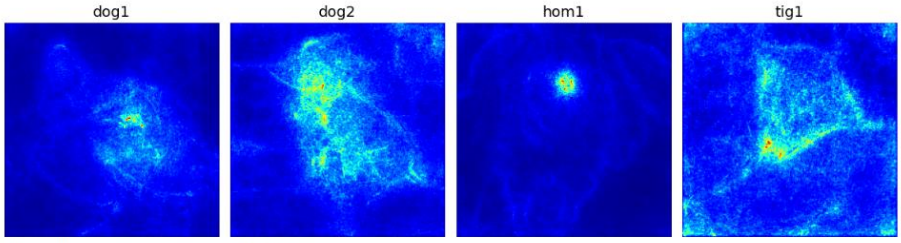
4. Благодаря Saliency сгенерируем карту внимания, подсвечивая входного изображения.

```
from tensorflow.keras import backend as K
from tf_keras_vis.saliency import Saliency

saliency = Saliency(model,
                    model_modifier=replace2linear,
                    clone=True)
saliency_map = saliency(score, X)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))
for i, title in enumerate(image_titles):
    ax[i].set_title(title, fontsize=16)
    ax[i].imshow(saliency_map[i], cmap='jet')
    ax[i].axis('off')
plt.tight_layout()
plt.show()
```



5. Удалим шум благодаря методу SmoothGrad

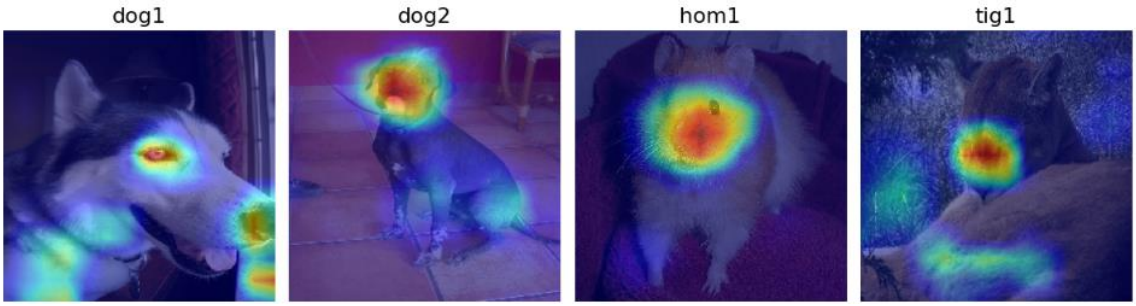
```
%%time


saliency_map = saliency(score,
                        X,
                        smooth_samples=20,
                        smooth_noise=0.20)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))
for i, title in enumerate(image_titles):
    ax[i].set_title(title, fontsize=14)
    ax[i].imshow(saliency_map[i], cmap='jet')
    ax[i].axis('off')
plt.tight_layout()
plt.savefig('smoothgrad.png')
plt.show()
```




```
CPU times: user 5min 12s, sys: 11.7 s, total: 5min 24s
Wall time: 3min 35s
```

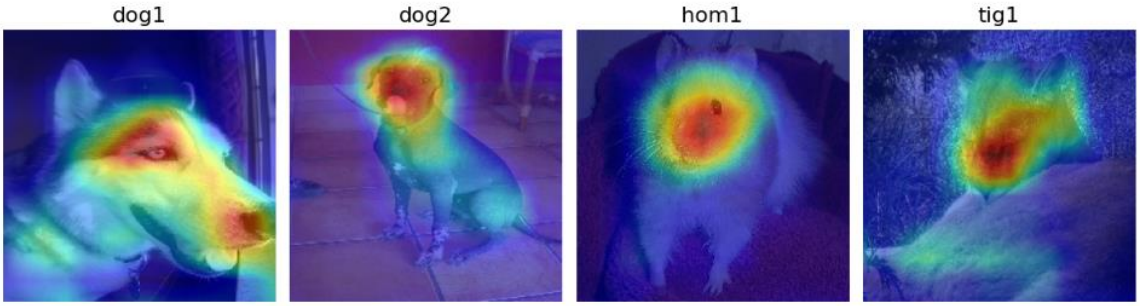
6. Попробуем способ генерации внимания GradCAM, которое проявляется в тех областях входного изображения, которые в наибольшей степени способствуют выходному значению

```
from matplotlib import cm
from tf_keras_vis.gradcam import Gradcam

gradcam = Gradcam(model,
                  model_modifier=replace2linear,
                  clone=True)
cam = gradcam(score,
              X,
              penultimate_layer=-1)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))
for i, title in enumerate(image_titles):
    heatmap = np.uint8(cm.jet(cam[i])[..., :4] * 255)
    ax[i].set_title(title, fontsize=16)
    ax[i].imshow(images[i])
    ax[i].imshow(heatmap, cmap='jet', alpha=0.5) # overlay
    ax[i].axis('off')
plt.tight_layout()
plt.show()
```



7. Попробуем также GradCAM++, который может предоставить прогнозы модели CNN

```
%%time

from tf_keras_vis.gradcam_plus_plus import GradcamPlusPlus

gradcam = GradcamPlusPlus(model,
                          model_modifier=replace2linear,
                          clone=True)

cam = gradcam(score,
              X,
              penultimate_layer=-1)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))
for i, title in enumerate(image_titles):
    heatmap = np.uint8(cm.jet(cam[i])[..., :4] * 255)
    ax[i].set_title(title, fontsize=16)
    ax[i].imshow(images[i])
    ax[i].imshow(heatmap, cmap='jet', alpha=0.5)
    ax[i].axis('off')
plt.tight_layout()
plt.savefig('gradcam_plus_plus.png')
plt.show()
```




```
CPU times: user 19.1 s, sys: 3.6 s, total: 22.7 s
Wall time: 16.9 s
```

Выводы

В ходе выполнения лабораторной работы можно сделать вывод, что более полным и точным методом описания активаций слоев является метод Guided backpropogation. Если стоит задача на устранение неинформативных областей, используемый метод отличен для визуализации.