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

ИКБ направление «Кибербразведка и противодействие угрозам с применением технологий искусственного интеллекта» 10.04.01

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Лабораторная работа №3

по дисциплине

«Анализ защищённости систем искусственного интеллекта»

Группа:
ББМО-01-22
Выполнила:
Огольцова Н.Д.

Проверил:
Спирин А.А.

Москва 2023

Ход работы:

К сожалению, так и не получилось выполнить представленный в задании код, поэтому на его основе создан тот, который представлен ниже.

Загрузим необходимые библиотеки.

```
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 1.8 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 1 GPUs
```

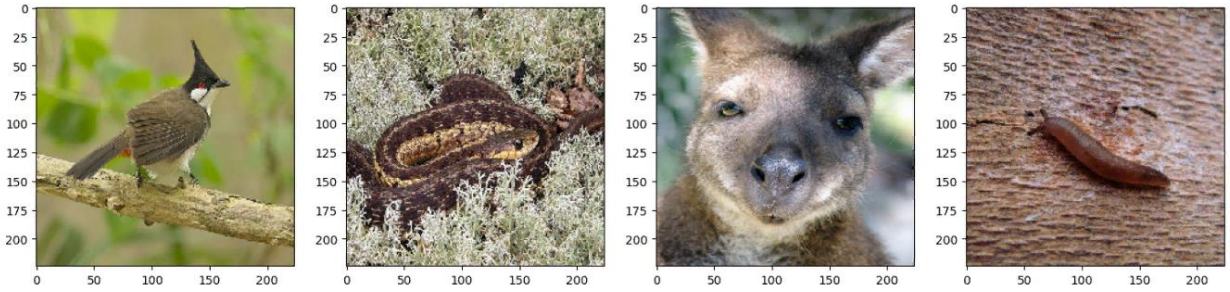
Выполним загрузку модели.

```
:  from tensorflow.keras.applications.vgg16 import VGG16 as Model
   model = Model(weights='imagenet', include_top=True)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_k
ernels.h5
553467096/553467096 [==============================] - 3s 0us/step
```

Загрузим и выполним предобработку 4 изображений и выведем результат.

```
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.applications.vgg16 import preprocess_input
from matplotlib import pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (18, 6)

img1 = load_img('/content/1.JPEG', target_size=(224, 224))
img2 = load_img('/content/2.JPEG', target_size=(224, 224))
img3 = load_img('/content/3.JPEG', target_size=(224, 224))
img4 = load_img('/content/4.JPEG', target_size=(224, 224))

f, ax = plt.subplots(1, 4)
ax[0].imshow(img1)
ax[1].imshow(img2)
ax[2].imshow(img3)
ax[3].imshow(img4)

names = ['1', '2', '3', '4']
images = np.asarray([np.array(img1), np.array(img2), np.array(img3), np.array(img4)])
X = preprocess_input(images)
```

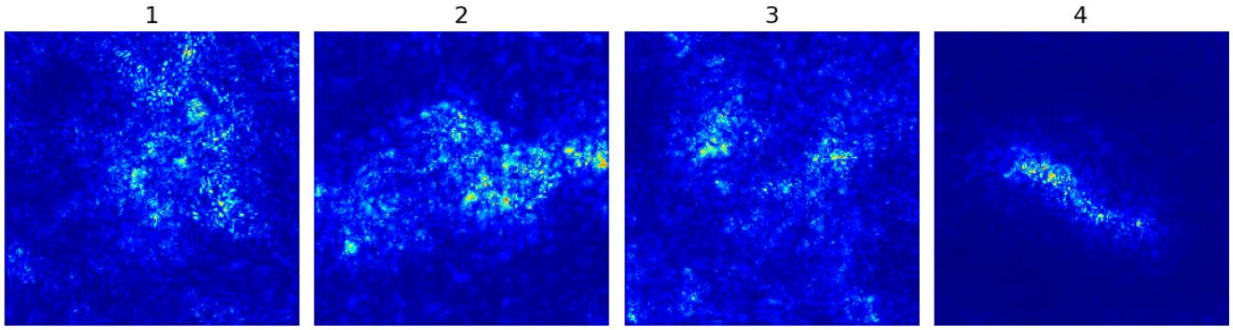


Заменим функцию на функцию линейной активации. И создадим функцию по подсчёту очков соответствия каждого изображения определённой группе.

```
from tf_keras_vis.utils.model_modifiers import ReplaceToLinear
replace2linear = ReplaceToLinear() # заменяем функцию на лицейную функцию активации
def model_modifier_function(cloned_model):
    cloned_model.layers[-1].activation = tf.keras.activations.linear


from tf_keras_vis.utils.scores import CategoricalScore
score = CategoricalScore([14, 24, 34, 44])
def score_function(output):
    return (output[0][14], output[1][24], output[2][34], output[3][44])
```

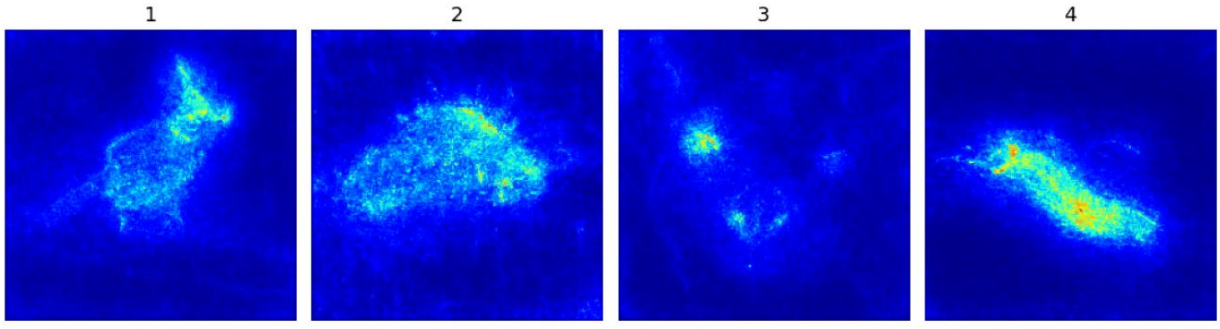
Сгенерируем карту внимания (vanilla), подсветим области наибольшего внимания.

```
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(names):
    ax[i].set_title(title, fontsize=16)
    ax[i].imshow(saliency_map[i], cmap='jet')
    ax[i].axis('off')
plt.tight_layout()
plt.show()
```



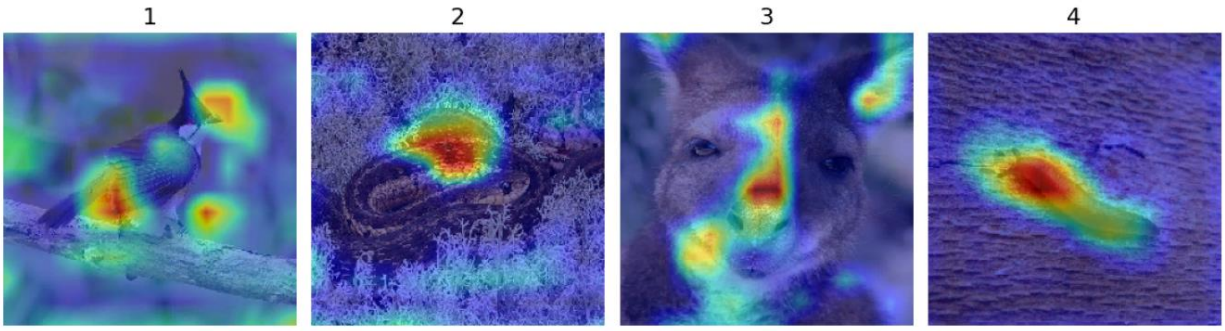
Уменьшим шум для карт влияния, с помощью SmoothGrad. И выведем результат.

```
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(names):
    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()
```



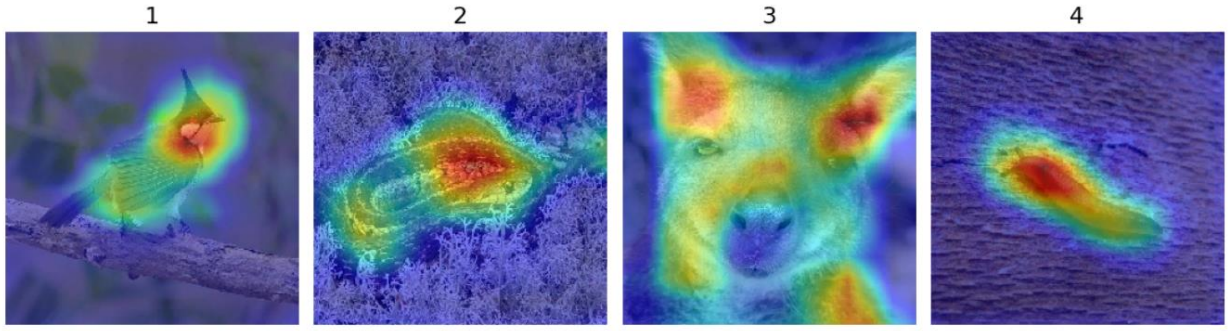
Сравним полученные значения с функцией GradCAM, визуализирует выходные данные свёрточного слоя.

```
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(names):
    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.show()
```



Сравним также с методом GradCAM++, который имеет ещё более лучшую визуализацию.

```
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(names):
    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()
```



Выводы:

В данной работе была исследована карта внимания, которая позволять понять работу обученной модели по категоризации изображения.

Использование метода SmoothGrad позволило уменьшить шумы на карте, в следствие чего были получены более точные результаты карт внимания, с более обширными выявленными областями областями.

Также было проведено сравнение методов GradCAM и GradCAM++. Данные карты внимания основаны на выводе предпоследнего свёрточного слоя и позволяют получать градиентные результаты с точечными областями наибольшего влияния на результат.

Отличие GradCAM и GradCAM++ заключается в усовершенствовании модели, за счёт математический доработок, на результирующих изображениях можно увидеть, как GradCAM++ выделяет более полные и точные контура животных, чем GradCAM.