МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

Институт кибербезопасности и цифровых технологий

ЛАБОРАТОРНОЕ ЗАНЯТИЕ № 3

по дисциплине

«Анализ защищенности систем искусственного интеллекта»

Выполнил:

ББМО–01–22

Чадов В. Т.

Проверил:

Спирин А. А.

«Зачтено» «__»_____2023 г. _____

Москва 2023

Добавим требующиеся библиотеки, процесс показан на рисунке 1. А также модель VGG16, показана на рисунках 2-3.

```
# добавим библиотеки, установим tf-keras-vis
!pip install tf-keras-vis
%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
from tensorflow.keras.applications.vgg16 import VGG16 as Model
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.applications.vgg16 import preprocess_input
from tf_keras_vis.utils.model_modifiers import ReplaceToLinear
from tf_keras_vis.utils.scores import CategoricalScore
from tensorflow.keras import backend as K
from tf_keras_vis.saliency import Saliency
from matplotlib import cm
from tf_keras_vis.gradcam import Gradcam
from tf_keras_vis.gradcam_plus_plus import GradcamPlusPlus
```

```
Requirement already satisfied: tf-keras-vis in /usr/local/lib/python3.10/dist-packages (0.8.6)
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)
Requirement already satisfied: deprecated in /usr/local/lib/python3.10/dist-packages (from tf-keras-vis) (1.2.14)
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)
```

Рис.2. библиотеки

```
[5]  # загрузим модель VGG16

     model = Model(weights='imagenet', include_top=True)
     model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
```

Рис.2. модель VGG16

```
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рис.3. модель VGG16

Загрузим и предобработаем изображения.

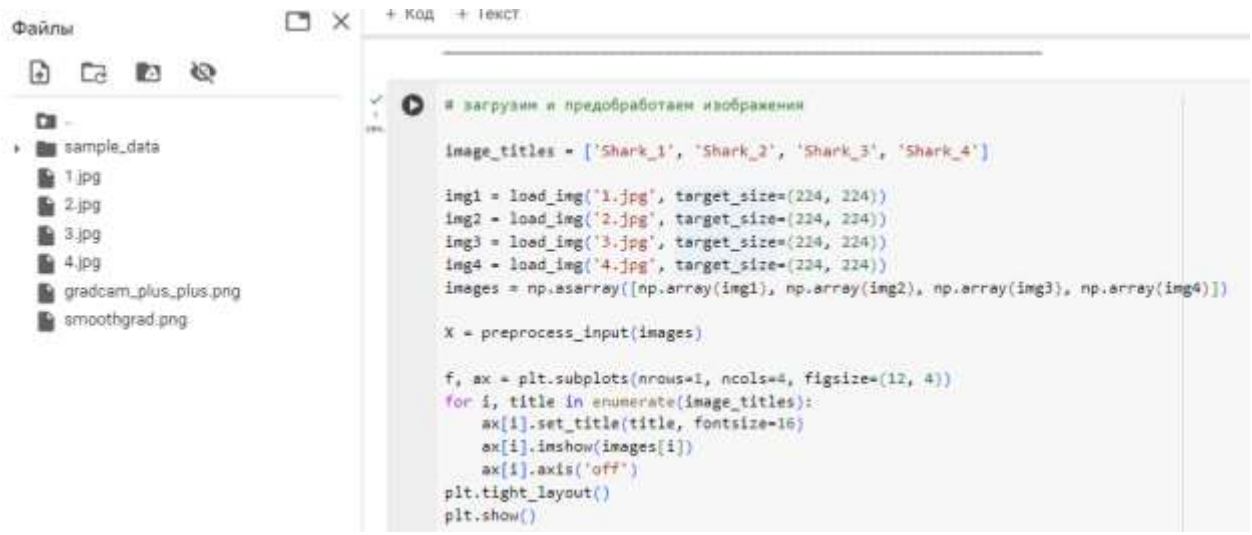


Рис.4. предобработка используемых изображений

Рис.5. используемые изображения

Реализуем функцию для линейной активации в последнем слое модели вместо softmax (улучшение созданий изображений внимания). А также функцию расчета score, в нашем случае 308 для акулы. Показаны на рисунке 6.

```python
replace2linear = ReplaceToLinear()

def model_modifier_function(cloned_model):
    cloned_model.layers[-1].activation = tf.keras.activations.linear

score = CategoricalScore([308, 308, 308, 308]) # 308 - акула

def score_function(output):
    return (output[0][308], output[1][308], output[2][308], output[3][308])
```

Рис.6. функция линейной активации и расчета score

Постром карты ванильного внимания. Видим низкое качество карты, непонятно для человеческого глаза. Показаны на рисунке 7.

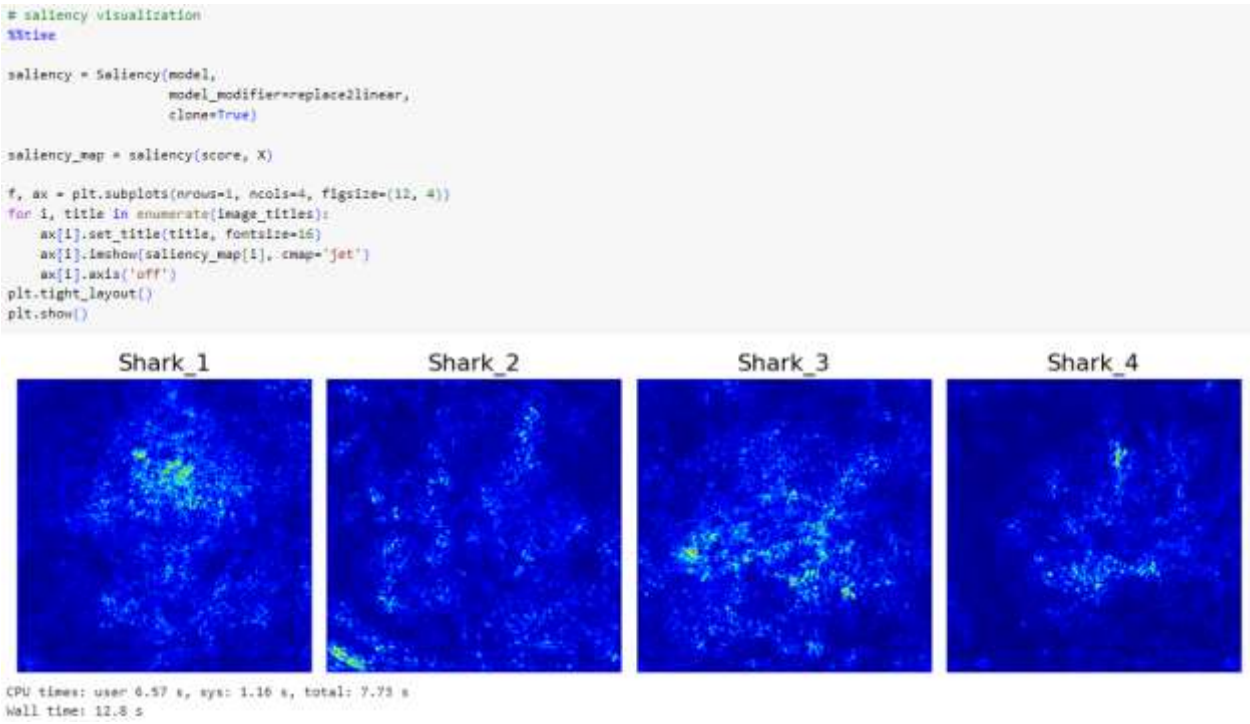


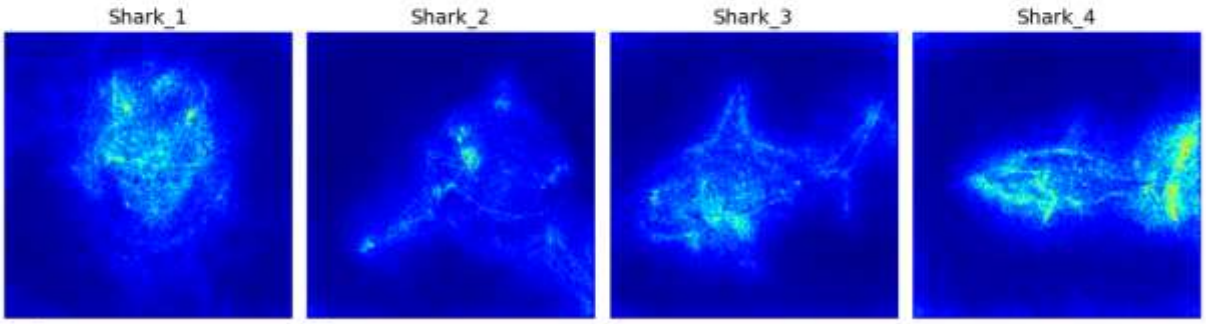
Рис.7. ванильное внимание

Постром карты smoothgrad. Видим улучшенное качество карты, можно понять что изначальный объект акула. Показаны на рисунке 8.

```
# карта saliency слишком шумная, отобразим карту с SmoothGrad
%%time

# уменьшим шум за счет добавления шума
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 2.27 s, sys: 314 ms, total: 2.59 s
Wall time: 3.93 s
```

Рис.8. smoothgrad

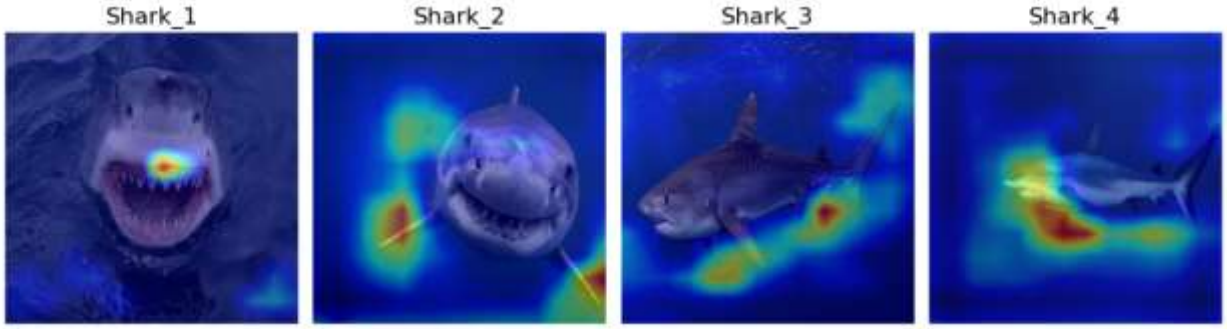
Попробуем способ gradcam. Изначальный обьект виден, но карта явно не охватывает основную цель изображения. Карты показаны на рисунке 9.

```
# отобразим способ Gradcam
%%time

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()
```




```
CPU times: user 1.52 s, sys: 856 ms, total: 2.38 s
Wall time: 2.26 s
```

Рис.9. gradcam

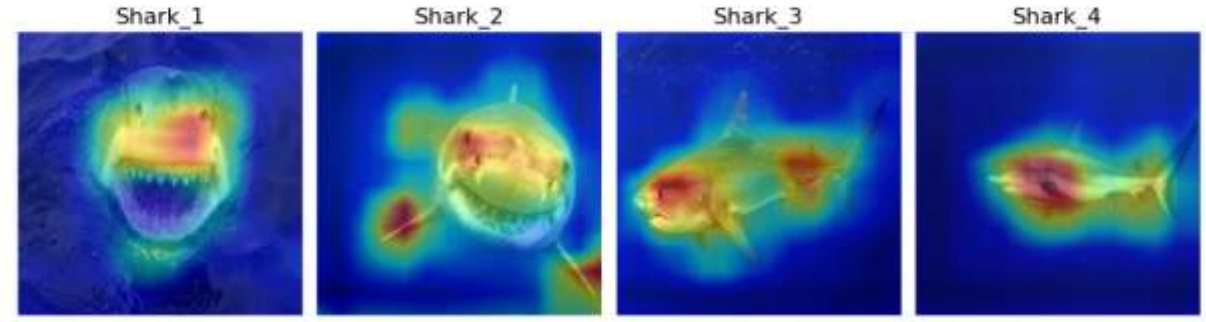
Отобразим gradcam++. Улучшенная версия gradcam полностью захватывает объект. Карты показаны на рисунке 10.

```
# отобразим улучшенный способ Gradcam++
%%time

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 2.82 s, sys: 849 ms, total: 2.67 s
Wall time: 2.82 s
```

Рис.10. gradcam++

Выводы

В лабораторной работе был разобран процесс построения карт внимания в нейронных сетях для анализа изображений из датасета ImageNet. В ходе работы были выполнены следующие шаги:

Замена функции активации softmax на линейную для корректного вычисления градиентов.

Построение карт значимости классов для выбранных изображений методами saliency, smoothgrad, gradcam, gradcam++.

Сравнение результатов и выводы о наиболее точном и полном методе описания активаций слоев нейронной сети.

В результате лабораторной работы были получены информативные карты значимости признаков и классов для изображений из датасета ImageNet. Это позволило лучше понять, какие части изображений влияют на классификацию, и освоиться с методами построения карт внимания.