МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Кафедра: КБ-4 «Кибербразведка и противодействие угрозам с применением технологий искусственного интеллекта»

Лабораторная работа

№3 по дисциплине

«Анализ защищенности систем искусственного интеллекта»

Выпол

ни: Морин

А.А.

Группа:

ББМО-01-23

Москва, 2024 г.

Добавим требующиеся библиотеки и установим keras, а также модель VGG16

[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 1.3 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
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from imageio->tf-keras-vis) (1.23
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

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

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

```
1 GPUs
```

```
from tensorflow.keras.applications.vgg16 import VGG16 as Model

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf
553467096/553467096 [==============================] - 20s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (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

 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)
_________________________________________________________________
```

Загрузим и подготовим изображения

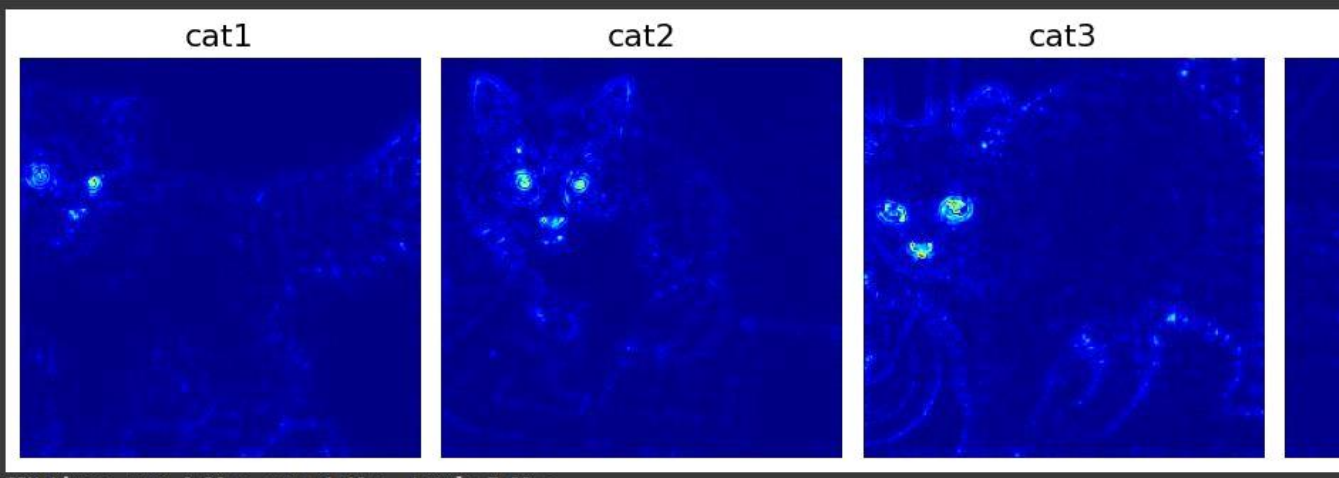
```
image_titles = ['cat1', 'cat2', 'cat3', 'cat4']

img0 = load_img('cat1.jpg', target_size=(224, 224))
img1 = load_img('cat2.jpg', target_size=(224, 224))
img2 = load_img('cat3.jpg', target_size=(224, 224))
img3 = load_img('cat4.jpg', target_size=(224, 224))
images = np.asarray([np.array(img0), np.array(img1), np.array(img2), np.array(img3)])

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()
```

Реализуем функцию для линейной активации в последнем слое модели вместо softmax (улучшение созданий изображений внимания). А также функцию расчета score, в нашем случае 284 для кота

```
[6]  from tf_keras_vis.utils.model_modifiers import ReplaceToL
     from tf_keras_vis.utils.model_modifiers import GuidedBack

     replace2linear = ReplaceToLinear()
     guided = GuidedBackpropagation()

[7]  from tf_keras_vis.utils.scores import CategoricalScore

     score = CategoricalScore([284, 284, 284, 284])

     def score_function(output):
         return (output[0][22], output[1][92], output[2][26],
```

Смотрим карты ванильного внимания. Видим низкое качество карты, коты уже вырисовываются, но пока слабо различимы.

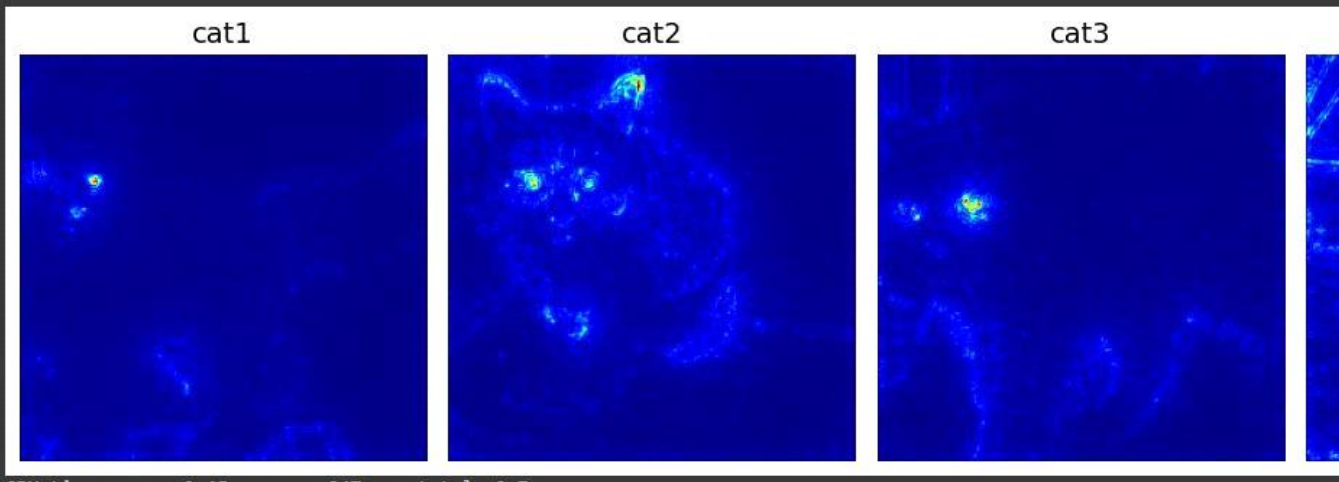
```
[8] %%time
    from tensorflow.keras import backend as K
    from tf_keras_vis.saliency import Saliency

    saliency = Saliency(model, model_modifier=guided, 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()
```




```
CPU times: user 6.86 s, sys: 1.12 s, total: 7.98 s
Wall time: 9.44 s
```

Смотрим карты smoothgrad. Видим улучшенное качество карты, можно понять, что изначальный объект кот.

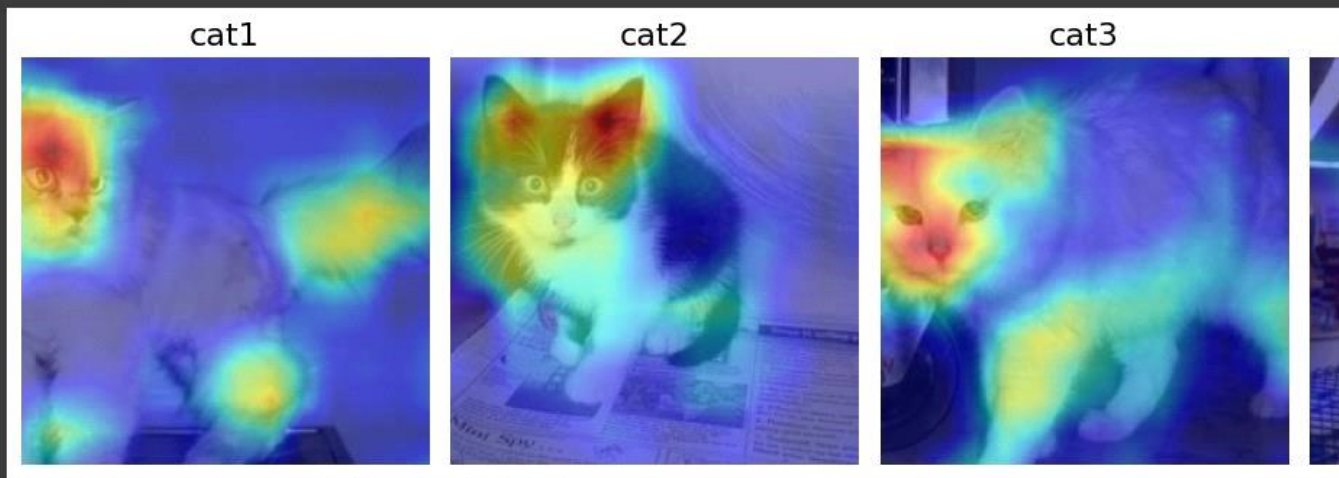
```
[9] %%time

    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 2.45 s, sys: 247 ms, total: 2.7 s
Wall time: 3.65 s
```

Попробуем способ gradcam. Изначальный обьект виден, но карта явно не охватывает основную цель изображения.

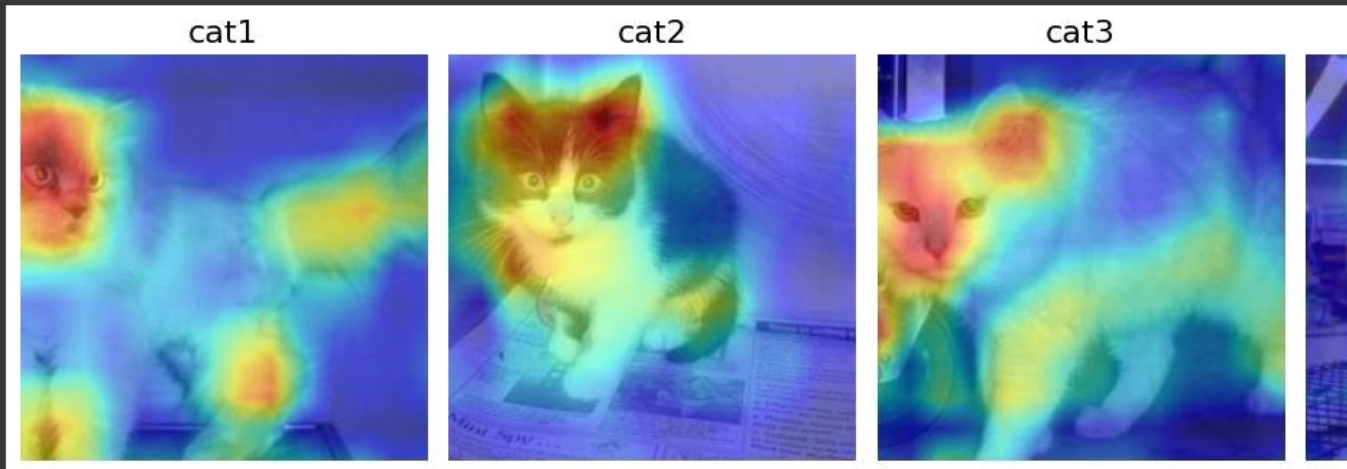
```
[10] %%time

     from matplotlib import cm
     from tf_keras_vis.gradcam import Gradcam

     gradcam = Gradcam(model, model_modifier=guided, 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])[..., :3] * 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 1.54 s, sys: 895 ms, total: 2.43 s
Wall time: 2.3 s
```

Отобразим gradcam++. Улучшенная версия gradcam практически полностью захватывает объект.

```
%%time

from tf_keras_vis.gradcam_plus_plus import GradcamPlusPlus

gradcam = GradcamPlusPlus(model, model_modifier=guided, 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])[..., :3] * 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 2.77 s, sys: 852 ms, total: 3.62 s
Wall time: 4.45 s
```

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.