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 защищенности систем искусственного интеллекта»

Группа:
ББМО-01-22
Выполнил:
Феденёв А.В.

Проверил:
Спирин А.А.

Москва 2023

Загружаем необходимые библиотеки.

```
[ ] !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 841.7 kB/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
```

```
[ ] %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))
    from tensorflow.keras.preprocessing.image import load_img
    from tensorflow.keras.applications.vgg16 import preprocess_input
```

```
Tensorflow recognized 1 GPUs
```

Выполним загрузку модели.

```
[ ] 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_dim_ordering_tf_kernels.h5
553467096/553467096 [==============================] - 25s 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
```

Загрузим и выполним предобработку изображений.

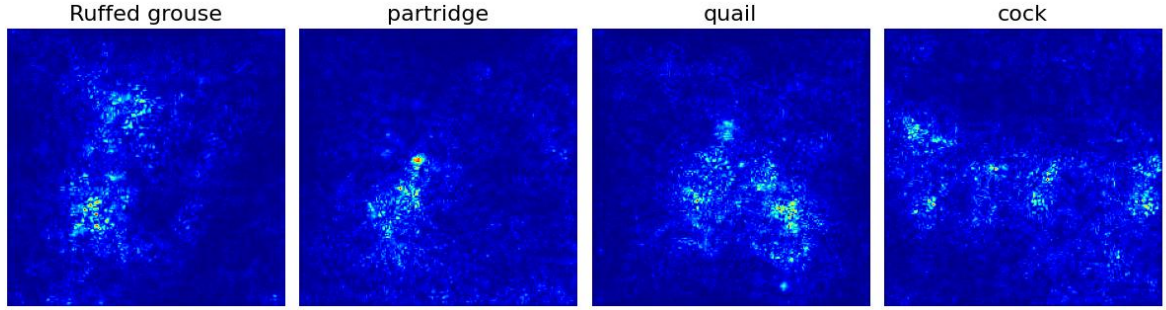
```
image_titles = ['Ruffed grouse', 'partridge', 'quail', 'cock']

img0 = load_img('Ruffed grouse.png', target_size=(224, 224))
img1 = load_img('partridge.png', target_size=(224, 224))
img2 = load_img('quail.png', target_size=(224, 224))
img3 = load_img('cock.png', 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()
```

Заменим на функцию линейной активации. Создадим функцию по подсчету очков соответствия изображений определенной группе.

```
[ ]  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

     from tf_keras_vis.utils.scores import CategoricalScore
     score = CategoricalScore([41, 42, 62, 63])
     def score_function(output):
         return (output[0][41], output[1][42], output[2][62], output[3][63])
```

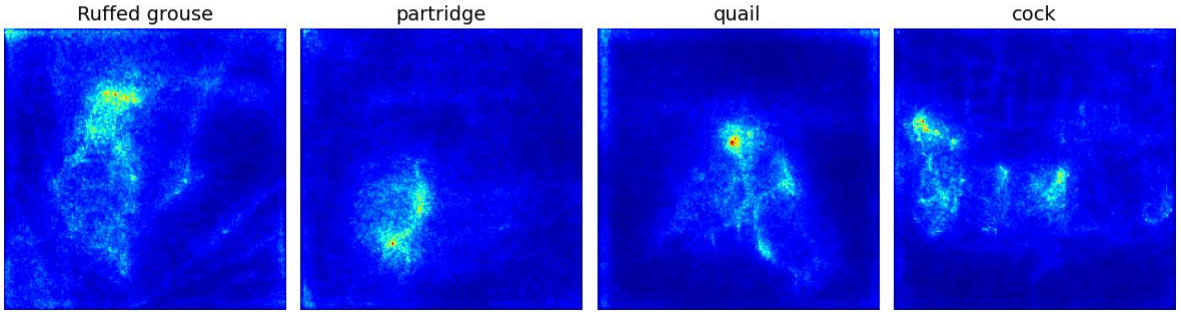
Сгенерируем карту внимания Vanilla.

```
[ ]  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()
```



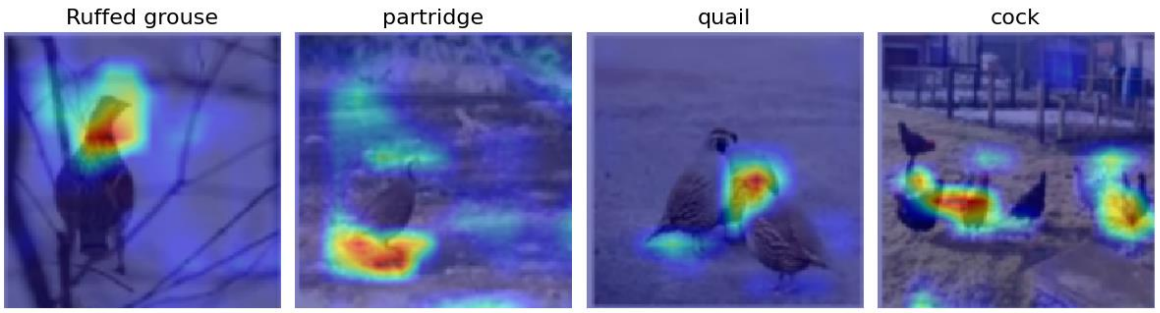
Уменьшим шум карт влияния с помощью SmoothGrad.

```
[ ]  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()
```



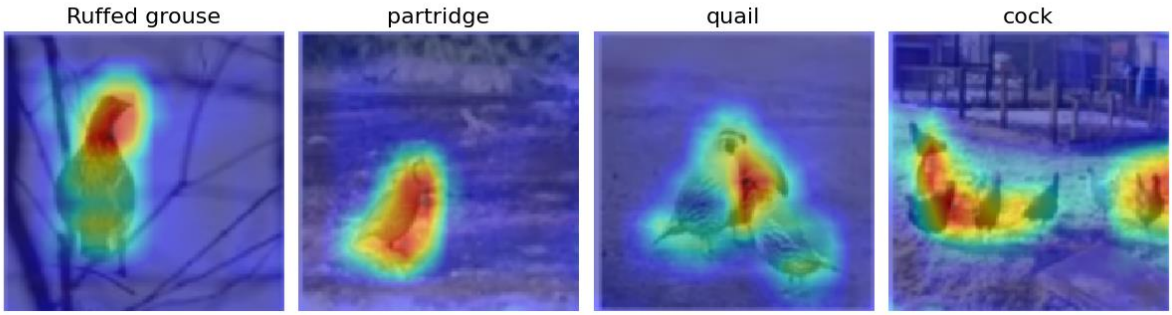
Сравним полученные значения с функцией GrandCAM, визуализирует выходные данные сверточного слоя.

```
[ ]  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(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()
```



Сравним также с методом GradCAM++, который имеет еще лучше 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(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()
```



SmoothGrad – техника, предназначенная для сглаживания карт выделенности с целью снижения шума и повышения интерпретируемости. GrandCAM – метод визуализации активации нейронов в сверточных нейронных сетях, который позволяет понять, какие участки входного изображения были наиболее значимыми для принятия окончательного решения моделью. GrandCAM++ - расширенный метод GrandCAM. Выбор между этими методами следует делать в зависимости от поставленной задачи.