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-02-22
Выполнил:
Запруднов
М.С.

Проверил:
Спирин А.А.

Москва 2023

Загрузим модель.

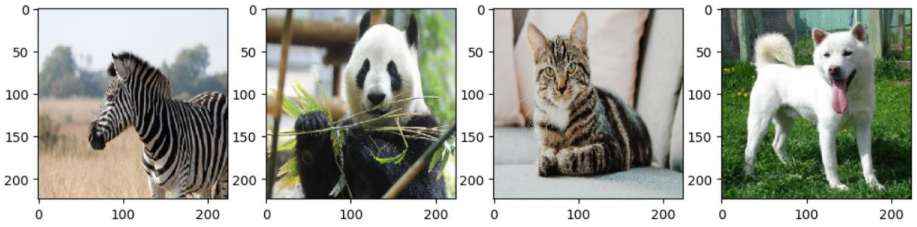
```
from tensorflow.keras.applications.vgg16 import VGG16 as Model
model = Model(weights='imagenet', include_top=True)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [==============================] - 21s 0us/step
```

Загрузим и выполним предобработку 4 изображений.

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

image_titles = ['zebra', 'panda', 'cat', 'dog']
img1 = load_img('zebra.jpg', target_size=(224, 224))
img2 = load_img('panda.jpg', target_size=(224, 224))
img3 = load_img('cat.jpg', target_size=(224, 224))
img4 = load_img('dog.jpg', 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))
ax[0].imshow(img1)
ax[1].imshow(img2)
ax[2].imshow(img3)
ax[3].imshow(img4)
```

```
<matplotlib.image.AxesImage at 0x7954103e64a0>
```

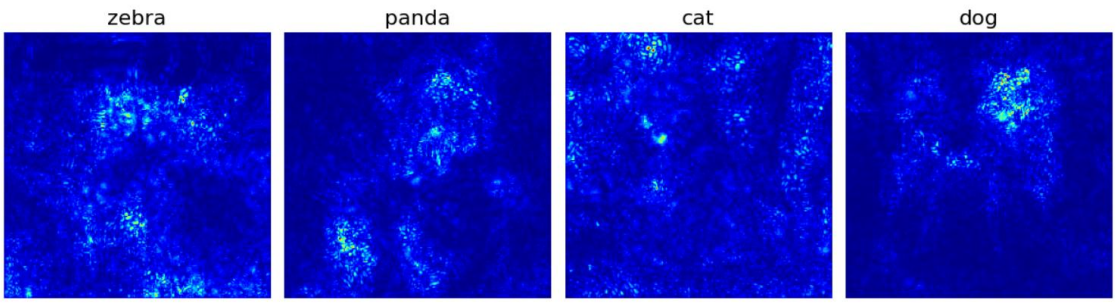


Заменим функцию на линейную функцию, а также создадим функциюпо подсчету очков соответствия каждого изображения определенной группе.

```
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([283, 153, 14, 673])
def score_function(output):
    return (output[0][8], output[1][18], output[2][23], output[3][31])
```

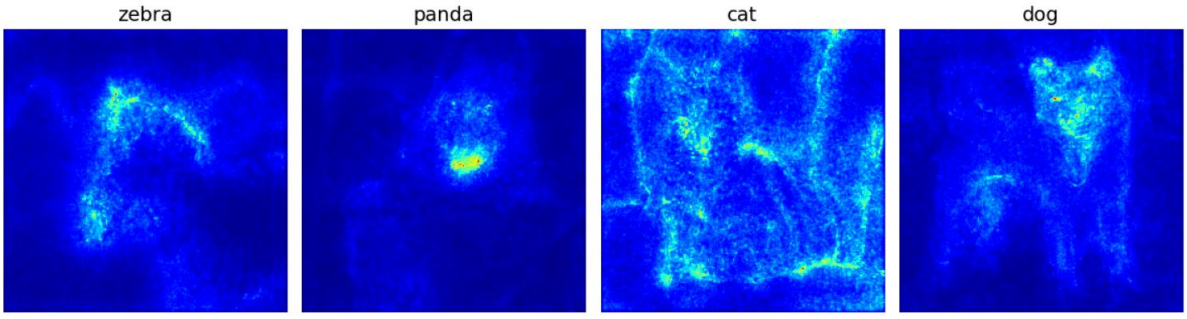
Сгенерируем карту внимания и подсветим области наибольшего внимания.

```
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(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()
```



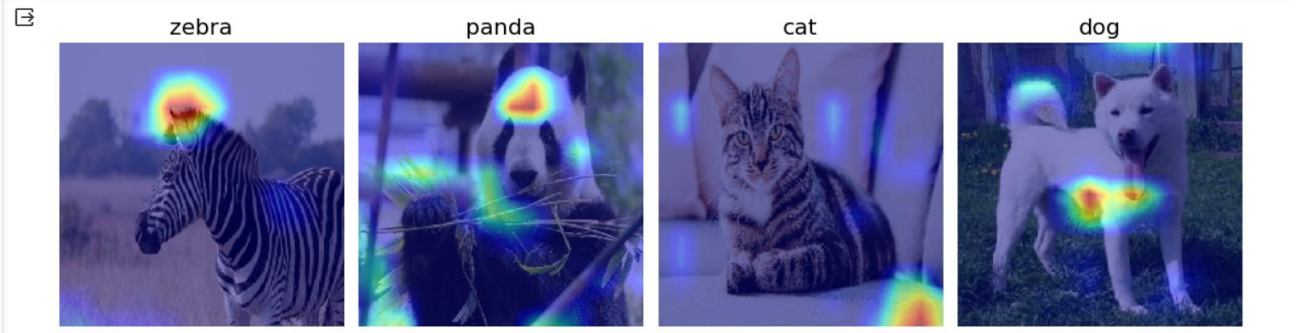
Уменьшим шум для карт влияния.

```
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()
```



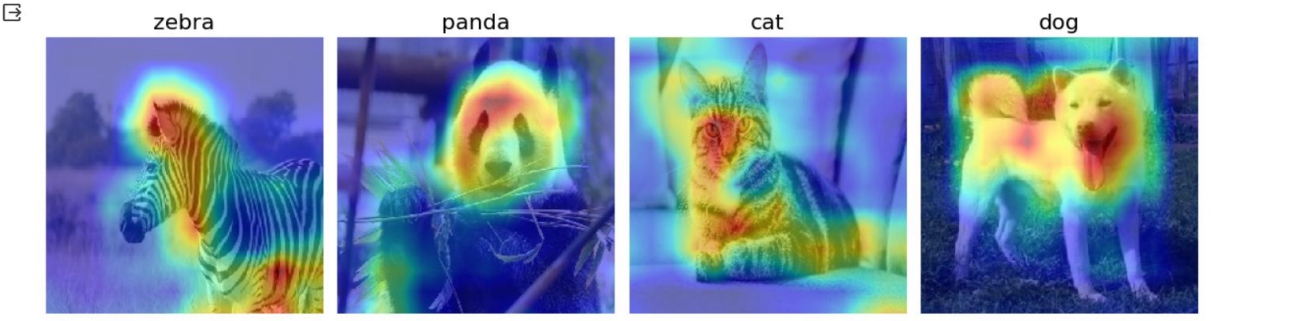
Сравним полученные значения с функцией GradCAm.

```
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 с методом GradCAM++.

```
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()
```



Выводы:

SmoothGRAD, GradCAM и GradCAM++ - это алгоритмы визуализации, которые помогают понять, какие области изображения влияют на решения нейронных сетей.

SmoothGRAD: Этот метод предлагает усреднение градиентов, добавляя случайный шум к входным данным. Это помогает сгладить изображение и уменьшить влияние шума, что делает визуализацию более интерпретируемой.

GradCAM (Gradient-weighted Class Activation Mapping): GradCAM фокусируется на активации конкретного класса, взвешивая градиенты, проходящие через слои сети. Это позволяет выделить важные части изображения, влияющие на принятое решение.

GradCAM++: Это улучшенная версия GradCAM, которая учитывает не только положительные, но и отрицательные влияния активаций при создании карт

активации. Это способствует более точной локализации важных областей на изображении.