



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ИКБ направление «Киберразведка 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 системы информационной
безопасности”

Группа:

ББМО-01-2

2

Выполнил

: Челышев Д.И.

Проверил:

Спирин А.А.

Москва 2023

Для начала установим инструмент для визуализации для TensorFlow Keras.

```
[ ] !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 1.1 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
```

Далее подключим необходимые библиотеки, а также активируем перезагрузку при изменении кода, чтобы не нужно было явно выполнять перезагрузку

```
[ ] %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
```

Загрузим предварительно обученную модель VGG16, на ImageNet датасете. После чего отобразим сводку по 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 [=====] - 3s 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 несколько изображений датасета ImageNet и выполним их предварительную обработку перед использованием.

Отобразим на одном графическом представлении все наши изображения.

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

image_titles = ['1', '2', '3', '4']

img0 = load_img('1.jpg', target_size=(224, 224))
img1 = load_img('2.jpg', target_size=(224, 224))
img2 = load_img('3.jpg', target_size=(224, 224))
img3 = load_img('4.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()
```



Заменяем функцию активации на линейную

```
[ ] 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
```

Создадим функцию очков соответствия каждому изображению

```
[ ] from tf_keras_vis.utils.scores import CategoricalScore
score = CategoricalScore([11, 12, 13, 14])
def score_function(output):
    return (output[0][11], output[1][12], output[2][13], output[3][14])
```

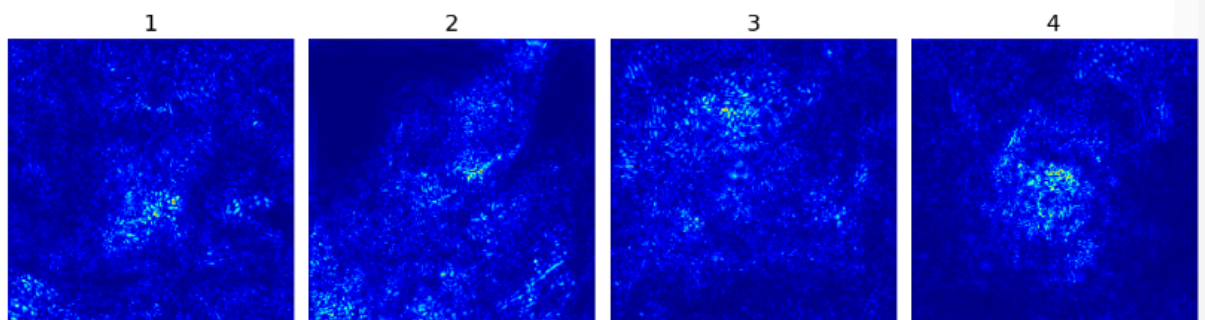
Создадим карту внимания (vanilla)

```
from tf_keras_vis.saliency import Saliency

saliency = Saliency(model,model_modifier=replace2linear,clone=True)

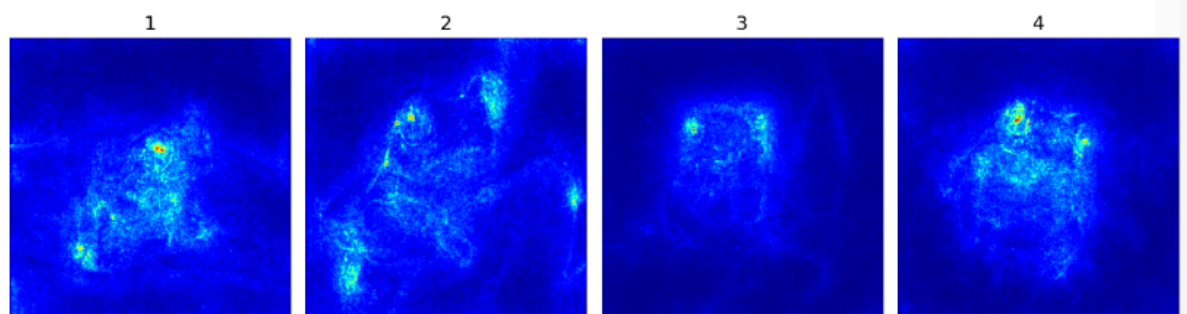
saliency_map = saliency(score, X)

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 для карт влияния

```
[8] %%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 2.36 s, sys: 287 ms, total: 2.64 s

Используем GrandCAM

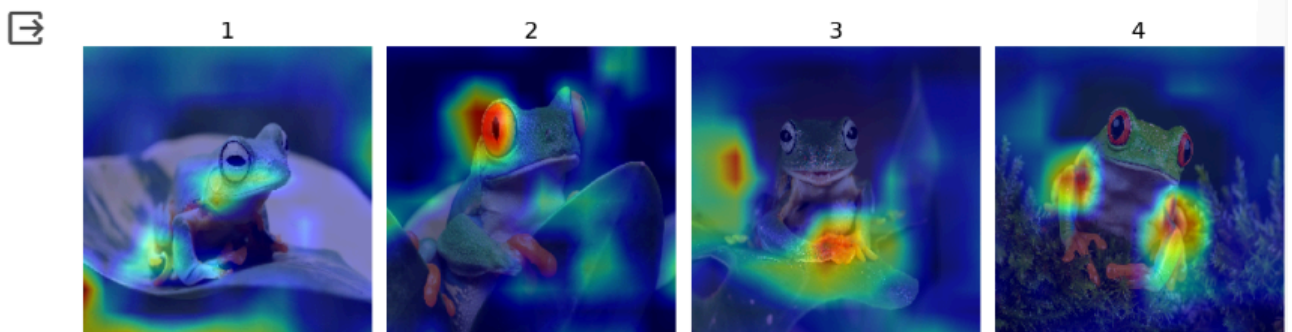
```

▶ 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()

```



Используем GradCAM++

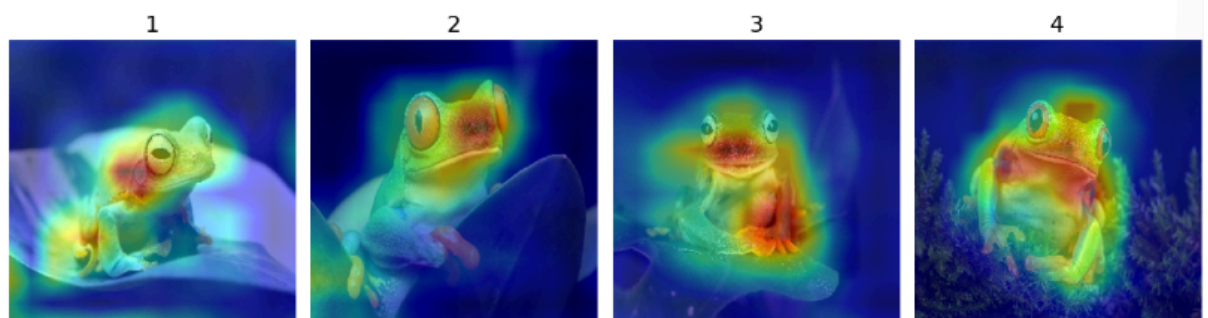
```

▶ 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()

```



Вывод

Таким образом, использование методов визуализации Grad-CAM, Grad-CAM++, Saliency, SmoothGrad и подобных, может быть полезным для понимания того, какие части изображений были наиболее важными при принятии решений моделью машинного обучения. Методы Grad-CAM и Grad-CAM++ позволяют визуализировать активации в различных частях изображений, позволяя понять, где модель фокусируется при определении

классов. Карты активаций могут помочь исследовать, какие объекты или части изображения были ключевыми для принятия решения моделью. Использование метода SmoothGrad может помочь снизить шум и сделать карты активации более интерпретируемыми