



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

---

Институт кибербезопасности и цифровых технологий  
КБ-4 «Интеллектуальные системы информационной безопасности»

Отчет по лабораторной работе №3  
по дисциплине: «Анализ защищенности систем искусственного  
интеллекта»

**Выполнил:**

Студент группы ББМО-02-22  
Филиппов Леонид Алексеевич

**Проверил:**

Спирин Андрей Андреевич

Москва 2024

## 1. Установка библиотек

```
1. Установка библиотек

[2] !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 1.5 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
```

## 2. Определение кол-ва GPU

```
2. Определение кол-ва GPU

[22] %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.utils import load_img

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

Tensorflow recognized 1 GPUs
```

## 3. Загрузка модели

```
3. Загрузка модели

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

## 4. Загрузка и предобработка 4 изображений

```
4. Загрузка и предобработка 4 изображений

[23] from google.colab import drive

drive.mount('/content/drive')

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


image_titles = ['avocado', 'goy', 'pivo', 'tyler']
img1 = load_img('/content/drive/My Drive/avocado.jpg', target_size=(224, 224))
img2 = load_img('/content/drive/My Drive/goy.jpg', target_size=(224, 224))
img3 = load_img('/content/drive/My Drive/pivo.jpg', target_size=(224, 224))
img4 = load_img('/content/drive/My Drive/tyler.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)
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).



## 5. Замена функции на линейную функцию, а также создания функции по подсчету очков соответствия каждого изображения определенной группе

```
5. Замена функции на линейную функцию, а также создание функции по подсчету очков соответствия каждого изображения определенной группе.

[24] 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])
```

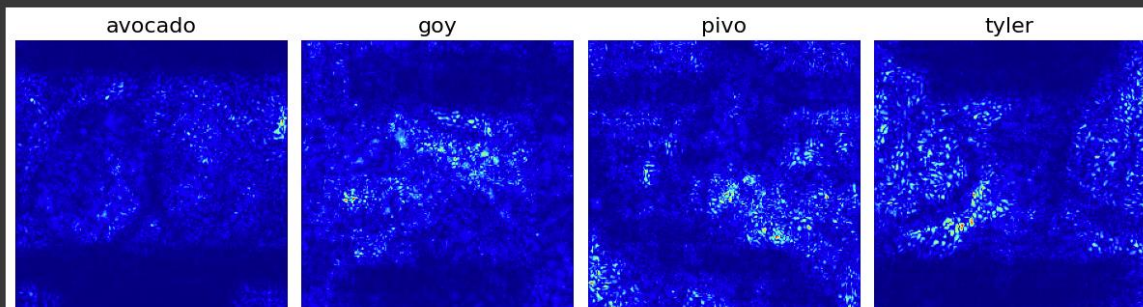
## 6. Генерация карты внимания и подсветка области наибольшего внимания

6. Генерация карты внимания и подсветка области наибольшего внимания.

```
✓ 12 OK. [25] 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()
```

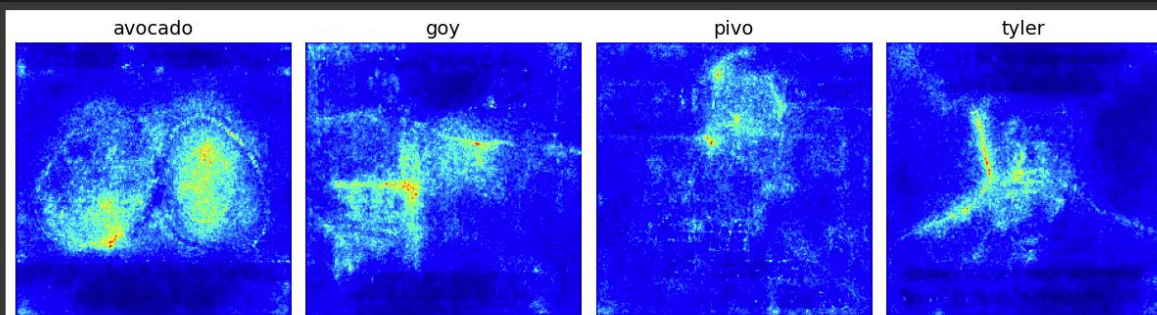


## 7. Уменьшение шума для карт влияния

7. Уменьшение шума для карт влияния

```
✓ 5 OK. [26] 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()
```





## 8. Сравнение полученных значений с функцией GradCam

### 8. Сравнение полученных значений с функцией GradCam

```
[27] 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()
```



## 9. Сравнение с методом GradCam++

### 9. Сравнение с методом GradCam++

```
[28] 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**

SmoothGRAD, GradCAM и GradCAM++ представляют собой методы визуализации, предназначенные для анализа воздействия различных областей изображения на решения нейронных сетей.

SmoothGRAD использует метод усреднения градиентов с добавлением случайного шума к входным данным. Это прием помогает сгладить изображение и уменьшить воздействие шума, что делает визуализацию более интерпретируемой.

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

GradCAM++ представляет собой улучшенную версию GradCAM, которая учитывает как положительные, так и отрицательные влияния активаций при создании карт активации. Это способствует более точной локализации значимых областей на изображении.