|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по дисциплине

«Анализ защищенности систем искусственного интеллекта»

**«Лабораторная работа 2»**

Выполнил студент группы ББМО-01-22

*Лысак Я.Д.*

Проверил

*Спирин Андрей Андреевич*

«27» ноября 2023 г.

Москва, 2023 г.

## Цель работы

Выполнить задания по построению карт активации различными способами и методами. Провести анализ полученных результатов, изменяя параметры и сделать соответствующие выводы по проделанной работе.

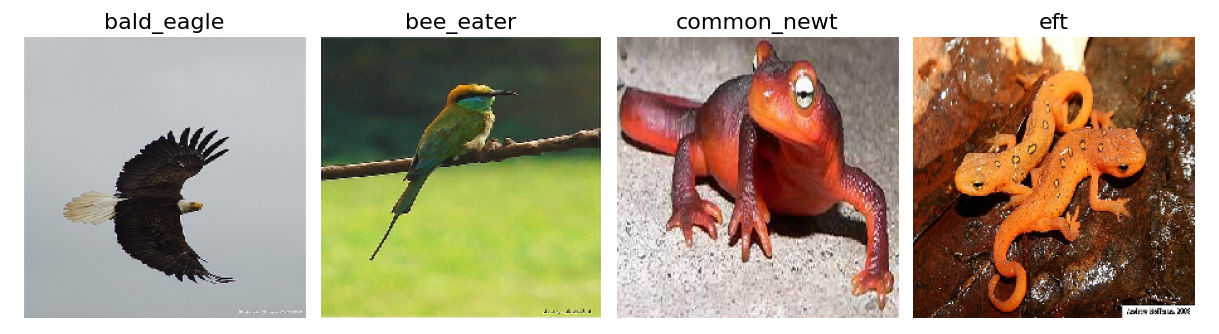
## Задание

1. Настроить и провести эксперимент по генерации карт значимости признаков на наборе данных imagenet;
2. оценить результаты, провести дополнительные эксперименты, изменяя параметр backprop\_modifier;
3. выполнить построение карт значимости классов для выбранных изображений методом для интерпретирования моделей gradCAM;
4. cделать выводы о наиболее точном и полном методе описания активаций слоев нейронной сети.

## Ход выполнения работы

В качестве набора данных для данной работы выступают 4 изображения, полученные из тестового набора данных ImageNet для ознакомления. Выбраны были 4 следующих изображения (рисунок 1).

Рисунок – 1



Данные классы или метки имеют следующие индексы в наборе ImageNet:

bald eagle – 22 – белый орёл

bee eater – 92 – пчелоед

common newt – 26 – обыкновенный тритон

eft – 27 – саламандра

Индексы для данных изображений были получены из следубщей части документации по ссылке: <https://deeplearning.cms.waikato.ac.nz/user-guide/class-maps/IMAGENET/>.

При установке, запуске и настройке возникли сложности, вызванные несовместимостью версий используемых библиотек и их зависимостей, в частности недоступности версии SciPy <1.1 и ее несовместимость с Python 3.9. При подробном изучении данного вопроса было найдено следующее обсуждение на GitHub, где указано, что библиотека более не поддерживается и приложена ссылка на обновленную версию: <https://github.com/raghakot/keras-vis/issues/221>

Новая версия библиотеки имеет совместимые методы и функции с предыдущее, поэтому в данной работе будет использоваться она.

Местом проведения лабораторной работы была выбрана среда Google Collab из-за простоты подключения и доступности графического ускорителя для более быстрого расчет результатов.

Файлы перемещаются в среду с помощью графического интерфейса. Установка и настройка зависимостей приведена во фрагменте кода 1.

Фрагмент кода – 1

!pip install tf-keras-vis

import numpy as np

from matplotlib import pyplot as plt

%matplotlib inline

from tensorflow.keras.preprocessing.image import load\_img

from tensorflow.keras.applications.vgg16 import preprocess\_input

В данной работе также используется графическое ускорение среды Collab, подключение во фрагменте кода 2.

Фрагмент кода – 2

from tf\_keras\_vis.utils import num\_of\_gpus

\_, gpus = num\_of\_gpus()

print('Tensorflow recognized {} GPUs'.format(gpus))

После выполнения фрагмент кода 1 и 2 производится загрузка предварительно обученной модели VGG16 на наборе данных ImageNet (фрагмент 3). Выводится структура модели.

Фрагмент кода – 3

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

model = Model(weights='imagenet', include\_top=True)

model.summary()

Следующий фрагмент кода (фрагмент кода 4) демонстрирует вывод исходных изображений. Для удобной обработки данные вносятся в массив данных NumPy. Полученные изображения приведены на рисунке 1.

Фрагмент кода – 3

from tensorflow.keras.preprocessing.image import load\_img

image\_titles = ['bald\_eagle', 'bee\_eater', 'common\_newt', 'eft']

img0 = load\_img('images/bald\_eagle.jpg', target\_size=(224, 224))

img1 = load\_img('images/bee\_eater.jpg', target\_size=(224, 224))

img2 = load\_img('images/common\_newt.jpg', target\_size=(224, 224))

img3 = load\_img('images/eft.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()

Перед следующий этапе изменяются параметры сети и перестройки графика, а именно производится замена функции "softmax" на "linear" в методы. На последнем слое модели изменяется функция активации, что продемонстрировано во фрагменте кода 4. Также во фрагменте 4 указана функция из библиотеки TF Keras Vis для автоматического изменения данного свойства сети.

Фрагмент кода – 4

model.layers[-1].activation = tf.keras.activations.linear

from tf\_keras\_vis.utils.model\_modifiers import ReplaceToLinear

Затем требуется указать функцию определения значения с соответствующими индексами (фрагмент кода 5).

Фрагмент кода – 5

from tf\_keras\_vis.utils.scores import CategoricalScore

score = CategoricalScore([22, 92, 26, 27])

def score\_function(output):

return (output[0][22], output[1][92], output[2][26], output[3][27])

После дополнительных приготовлений возможно провести первичную генерацию карт значимости признаков с использованием функции tf\_keras\_vis.saliency (фрагмент 6).

Фрагмент кода – 6

%%time

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')

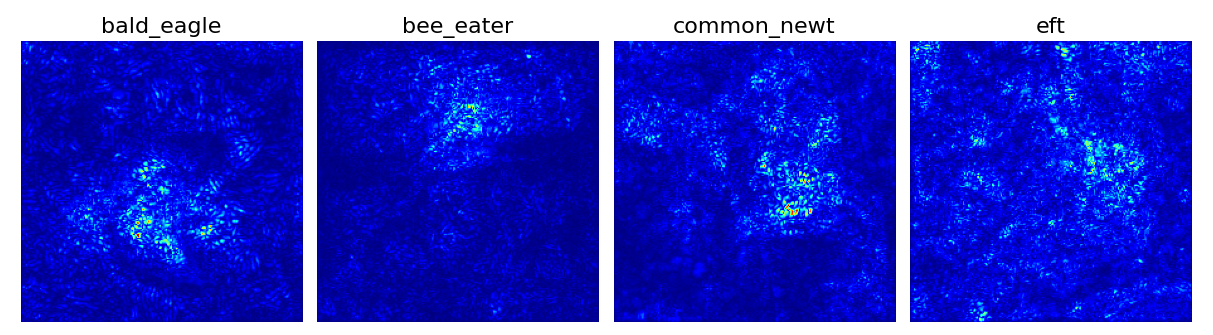
ax[i].axis('off')

plt.tight\_layout()

plt.show()

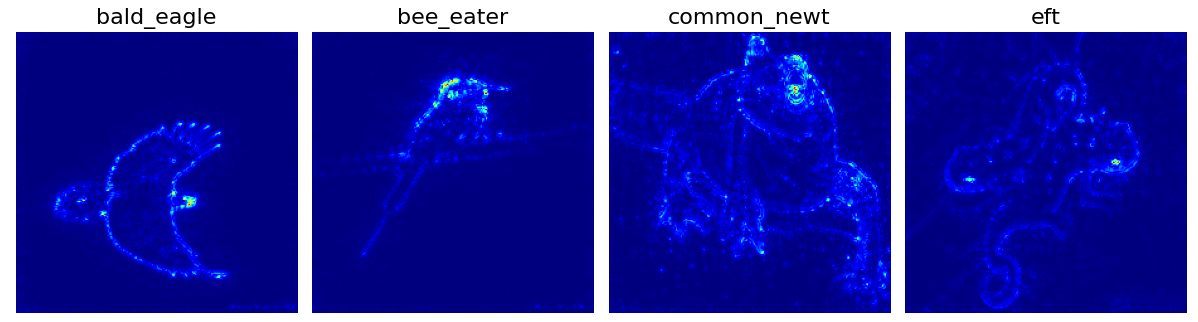
Полученные во фрагменте кода 6 результаты продемонстрированы на рисунке 2, где можно отметить, что результаты недостаточно точные, но демонстрируются некоторые аспекты и свойства определяемых объектов. Функция Saliency генерирует карту, на которой отображаются области входного изображения, вносящие наибольший вклад в выходное значение

Рисунок – 2



В рамках задачи также проводится эксперимент после смены модификатора с linear на guided, для данного примера приведен фрагмент кода 7. Демонстрация полученных результатов на рисунке 3.

Рисунок – 3



Фрагмент кода – 7

from tf\_keras\_vis.utils.model\_modifiers import ReplaceToLinear

from tf\_keras\_vis.utils.model\_modifiers import GuidedBackpropagation

replace2linear = ReplaceToLinear()

guided = GuidedBackpropagation()

saliency = Saliency(model, model\_modifier=guided, clone=True)

Нарисунке 4 продемонстрирован результаты обработки изображений методом GradCAM. Данный метод вместо использования градиентов выходных данных модели, использует выход предпоследнего слоя (то есть сверточного слоя, расположенного непосредственно перед плотными слоями). Фрагмент кода 8 содержит исполняет код для примеров, полученных на рисунке 4.

Фрагмент кода – 8

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])[..., :3] \* 255)

ax[i].set\_title(title, fontsize=16)

ax[i].imshow(images[i])

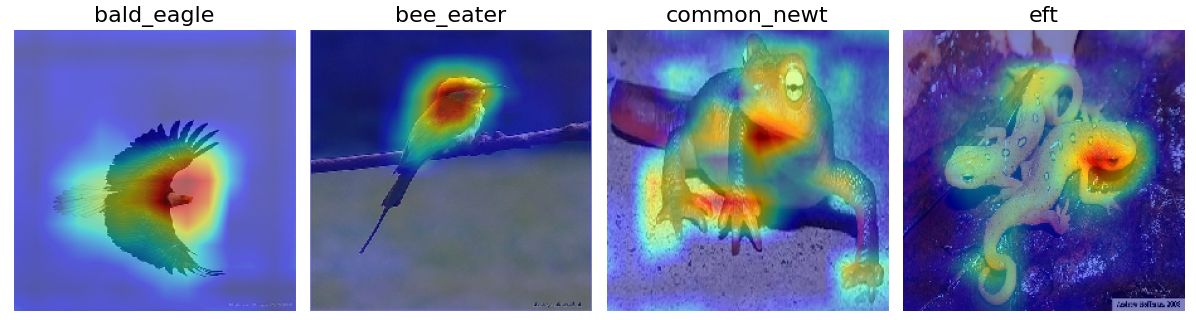
ax[i].imshow(heatmap, cmap='jet', alpha=0.5) # overlay

ax[i].axis('off')

plt.tight\_layout()

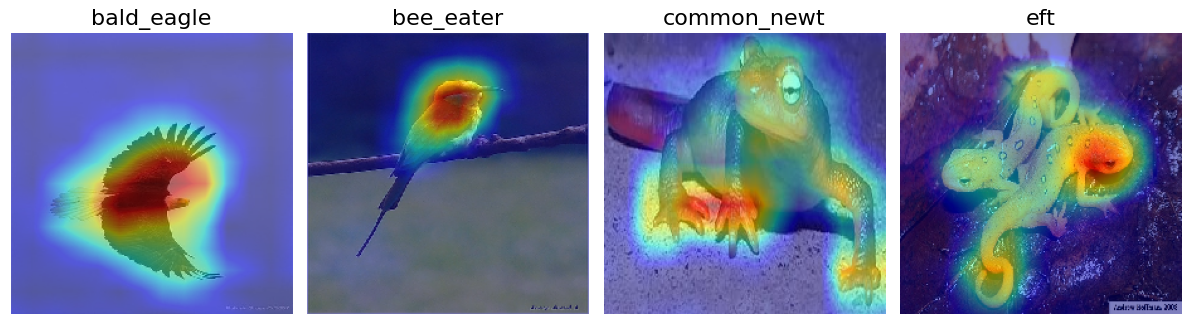
plt.show()

Рисунок – 4



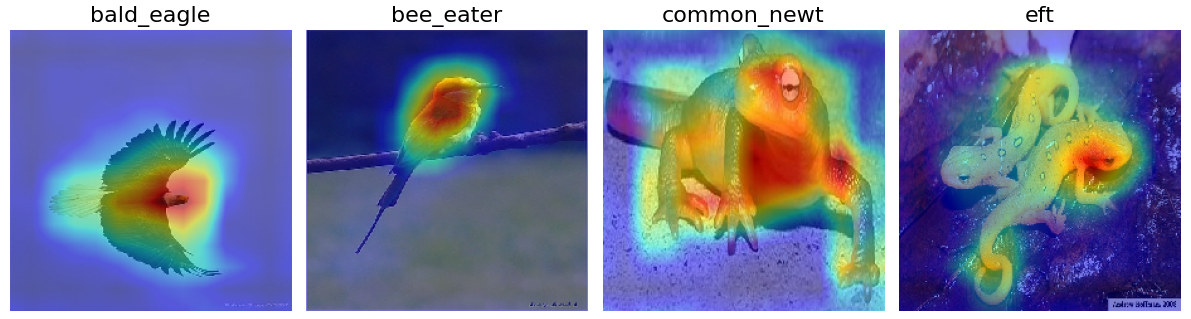
На рисунке 5 также продемонстрирован результат работы метода с модификатором guided.

Рисунок – 5



Также в данной библиотек есть и другие методы получения карты активации, например GradCAM++, пример результата которой приведен на рисунке 6. Как можно заметить результат несколько отличаются.

Рисунок – 6



На рисунке 7 отображен результат для метода GradCAM++ с модификацией guided.

Рисунок – 7



Как можно видеть GradCAM++ с модификацией guided демонстрирует, предположительно, наилучшие результаты среди всех участвующих методов и модификаций. На примере изображения с меткой common\_newt можно заметить, что данная модель берет в расчет не только части конечности земноводного, но также глаза и саму структуру конечности, а не только структуру фаланг.

Также видно и более точное определение региона на изображении, где сфокусированы данные, позволяющие точно определять тот или иной класс животного, как например с птицами.

**Вывод**

В результате выполнения работы получен некоторый опыт работы с инструментами получения карт активации для более подробного объяснения работы нейронных сетей и других моделей машинного обучения. Были также проведены эксперименты по созданию карты активации моделей машинного обучения с примерами на изображениях из набора данных ImageNet и модели VGG.

В рамках работы были рассмотрены несколько методов создания карт активации, в том числе GradCAM. Были проведены эксперименты с различными параметрами и модификациями методов.

После завершения экспериментов и проведения анализа полученных результатов были сделаны выводы касательного полноты и точности объяснения работы методов генерации карт.

Полный код блокнота по проведённой работе приведен в приложении А.

## Приложение А

!pip install tf-keras-vis

# %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))

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

model = Model(weights='imagenet', include\_top=True)

model.summary()

image\_titles = ['bald\_eagle', 'bee\_eater', 'common\_newt', 'eft']

img0 = load\_img('images/bald\_eagle.jpg', target\_size=(224, 224))

img1 = load\_img('images/bee\_eater.jpg', target\_size=(224, 224))

img2 = load\_img('images/common\_newt.jpg', target\_size=(224, 224))

img3 = load\_img('images/eft.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()

from tf\_keras\_vis.utils.model\_modifiers import ReplaceToLinear

from tf\_keras\_vis.utils.model\_modifiers import GuidedBackpropagation

replace2linear = ReplaceToLinear()

guided = GuidedBackpropagation()

from tf\_keras\_vis.utils.scores import CategoricalScore

score = CategoricalScore([22, 92, 26, 27])

def score\_function(output):

return (output[0][22], output[1][92], output[2][26], output[3][27])

%%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()

%%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('images/smoothgrad.png')

plt.show()

%%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()

%%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('images/gradcam\_plus\_plus.png')

plt.show()