



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

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

Отчёт по лабораторной работе №3

По дисциплине

«Анализ защищенности систем искусственного интеллекта»

Тема: «Использование механизмов внимания в нейронных сетях. Внимание в
CHC VGG (карта значимости признаков и grad-CAM)»

Студент Кузькин Павел Александрович

Группа БМО-01-22

Работу проверил

Спирин А.А.

Москва, 2023

Выполним установку tf-keras-vis:

```
!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 1.8 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
```

Использование Siliency для определения ключевых областей изображения

Загрузим предобученную модель VGG16:

```
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 [=====] - 3s 0us/step
```





Выберем 4 изображения из ImageNet и отобразим на одном полотне:

```
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np
from matplotlib import pyplot as plt
# Image titles
image_titles = ['hen', 'kite', 'loggerhead', 'tench']

# Load images and Convert them to a Numpy array
img1 = load_img('hen.JPEG', target_size=(224, 224))
img2 = load_img('kite.JPEG', target_size=(224, 224))
img3 = load_img('loggerhead.JPEG', target_size=(224, 224))
img4 = load_img('tench.JPEG', target_size=(224, 224))
images = np.asarray([np.array(img1), np.array(img2), np.array(img3), np.array(img4)])

# Preparing input data for VGG16
X = preprocess_input(images)

# Rendering
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()
```

hen	kite	loggerhead	tench
			

Заменим функцию активации на линейную:

```
from tf_keras_vis.utils.model_modifiers import ReplaceToLinear, GuidedBackpropagation
import tensorflow as tf

replace2linear = ReplaceToLinear()
guided = GuidedBackpropagation()

def model_modifier_function(cloned_model):
    cloned_model.layers[-1].activation = tf.keras.activations.linear
```

Установим классы изображений:

```
from tf_keras_vis.utils.scores import CategoricalScore

score = CategoricalScore([8, 21, 33, 0])
```

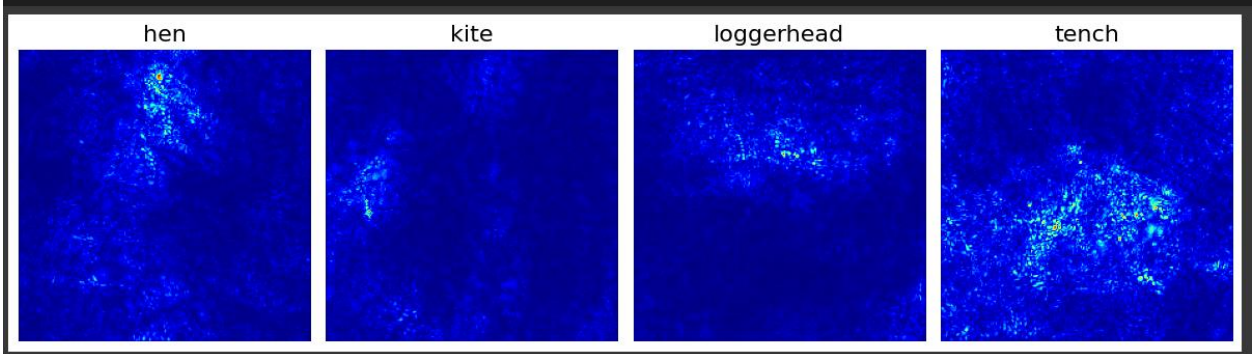
Отобразим стандартные ключевые области:

```
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()
```



Далее необходимо получить не стандартное отображение Saliency, а отображение с модифицированным градиентом активации (guided и rectified), при этом в отличие от старой версии устранение деконверсии (Rectified) не предусмотрено как отдельный метод, поэтому реализуем этот метод.

Rectified включает в себя уточнение обратного распространения для учёта активаций, которые становятся равными нулю в результате ReLU.

Вместо того, чтобы игнорировать эти нулевые активации, этот метод позволяет им внести вклад в градиенты через замену нулевых активаций на небольшие положительные значения.

```
from tf.keras_vis.utils.model_modifiers import ModelModifier
class RectifiedBackpropagation(ModelModifier):
    def __init__(self, target_activations=[tf.keras.activations.relu]) -> None:
        self.target_activations = target_activations

    def _get_rectified_activation(self, activation):
        @tf.custom_gradient
        def rectified_activation(x):
            def grad(dy):
                return tf.cast(dy > 0, dy.dtype) * dy

            return activation(x), grad

        return rectified_activation

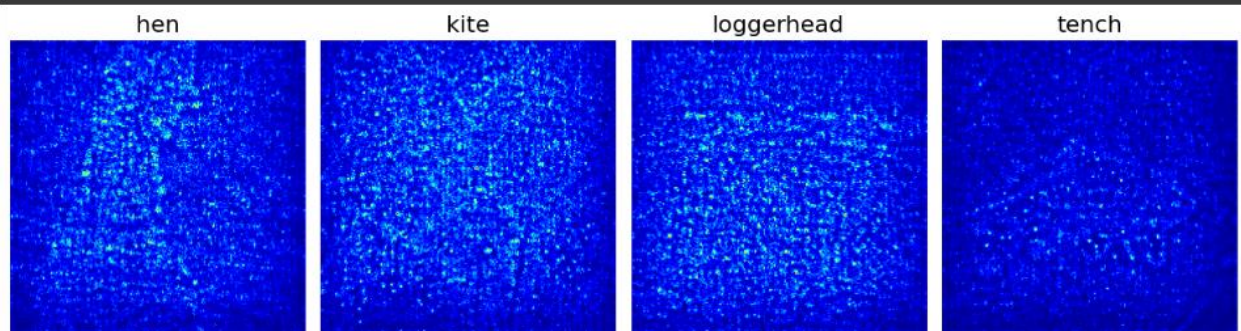
    def __call__(self, model) -> None:
        for layer in (layer for layer in model.layers if hasattr(layer, "activation")):
            if layer.activation in self.target_activations:
                layer.activation = self._get_rectified_activation(layer.activation)

rect = RectifiedBackpropagation()
from tf.keras_vis.saliency import Saliency

saliency = Saliency(model,
                    model_modifier=[replace2linear, rect],
                    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()
```



В **Gudied** при прохождении через узлы с функцией активации ReLU, градиенты сохраняются только для положительных значений входов, а для отрицательных устанавливаются в ноль.

```

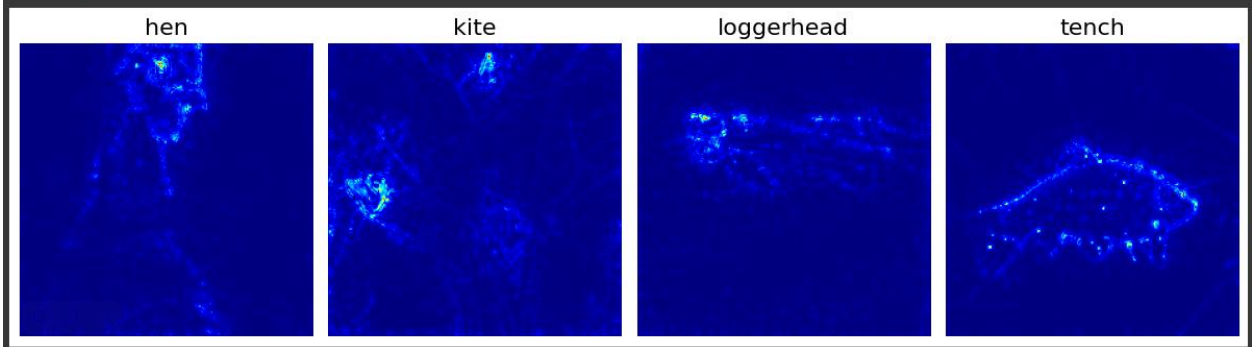
from tf_keras_vis.saliency import Saliency

saliency = Saliency(model,
                    model_modifier=[replace2linear, 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()

```



Необходимо также отметить, что преобразование градиентов `guided` помогает убрать "шум" из отображения активаций, в то время как `rectified` его только увеличивает, путём увеличения ненулевых значений. Для метода `Silience` модификатор `guided` имеет большое значение и позволяет чётче видеть области активаций.

Использование GradCAM для определения ключевых областей изображения

Получим краткое описание модели:

```
model.summary()
```

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)

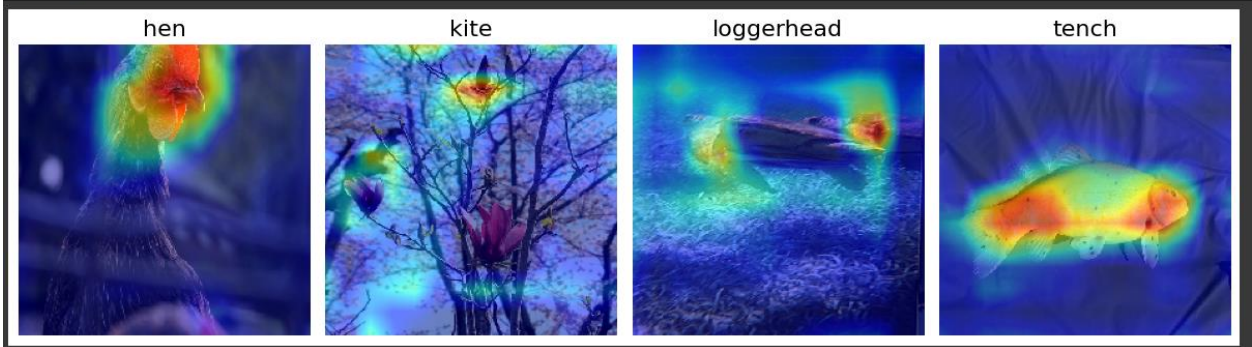
Выполним построение карт значимости классов для выбранных изображений методами vanilla:

```
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)
    ax[i].axis('off')
plt.tight_layout()
plt.show()
```

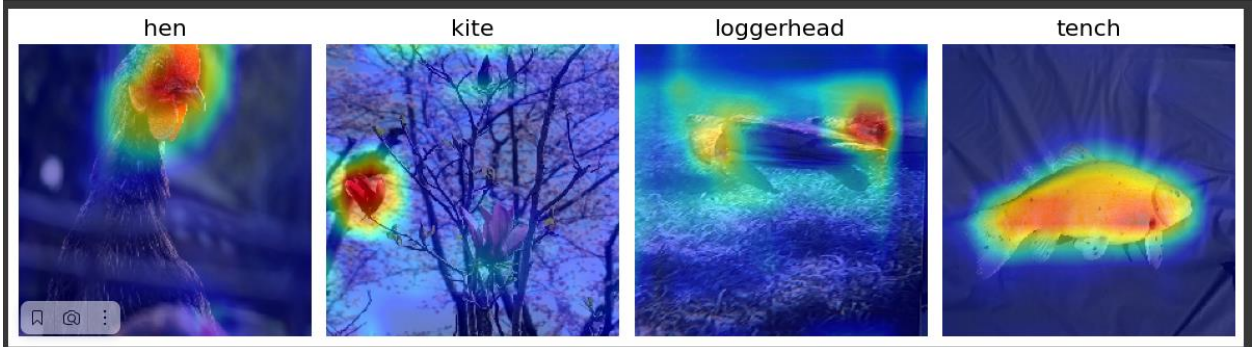


Выполним построение карт значимости классов для выбранных изображений методами guided:

```
gradcam = Gradcam(model,
                  model_modifier=[replace2linear, 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.show()
```



Выполним построение карт значимости классов для выбранных изображений методами rectified:



Таким образом, наиболее точным и полным методом описания активаций слоёв нейронной сети является guided backpropagation, который используется в области глубокого обучения, в частности, в контексте интерпретации и визуализации нейронных сетей. Его часто применяют для понимания и выделения важных областей входного изображения, которые влияют на принятие решения нейронной сети.

Стандартный алгоритм обратного распространения используется для обучения нейронных сетей путём вычисления градиентов по отношению к функции потерь и обновления весов в соответствии с ними. Guided Backpropagation, с другой стороны, модифицирует процесс обратного распространения, чтобы сосредотачиваться на положительных градиентах и подавлять отрицательные градиенты во время обратного прохода. Это помогает выделить области ввода, которые положительно влияют на выход нейронной сети.

Общий план метода Guided Backpropagation:

1) Прямой проход. Подается изображение на вход нейронной сети, выполняется прямой проход для вычисления выхода;

2) Обратный проход. Вычисляются градиенты выхода по отношению к входу во время обратного распространения;

3) Guided Backpropagation (во время обратного прохода для каждого нейрона или блока). Если градиент положительный, он передается дальше, если градиент отрицательный, устанавливается в ноль;

4) Визуализация. Модифицированные градиенты затем используются для выделения важных областей входного изображения.

Идея метода Guided Backpropagation заключается в фокусировке на положительных градиентах, так как они представляют области ввода, которые положительно влияют на активацию конкретного нейрона. Подавляя отрицательные градиенты, визуализация обычно выделяет черты, активирующие нейроны положительным образом, что упрощает интерпретацию и понимание того, какие части входного изображения влияют на решение модели.