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»

Отчет по лабораторной работе 3

по дисциплине
««Анализ защищенности систем искусственного интеллекта»»

Выполнил
Студент 2 курса: Стельмах Н.Е.
Группы: ББМО-02-22

Проверил:

Спирин А.А

Москва 2022 г.

Использование механизмов внимания в нейронных сетях

1. Внимание в CHC VGG (карта значимости признаков и grad-CAM)

a. Использовать репозиторий https://github.com/raghakot/kerasvis/blob/master/examples/vggnet/attention.ipynb

b. Чтобы визуализировать активацию на конечных выходах плотного слоя, необходимо переключить активацию softmax на линейную, поскольку градиент выходного узла будет зависеть от всех активаций других узлов. Сделать это в keras сложно, поэтому необходимо применить utils.apply_modifications для изменения параметров сети и перестройки графика. Если эта замена не будет выполнена, результаты могут быть неоптимальными. Мы начнем с замены "softmax" на "linear".

c. Загрузить 4 различных изображения из датасета ImageNet и отобразить их на одном графике с помощью команды: (img1 = utils.load_img('***.jpg/png', target_size=(224, 224)))

d. Отобразить карты значимости признаков, найти и задать верный параметр filter_indices, отображающий класс изображения ImageNet.

e. Оценить полученный результат на прошлом этапе и отображение признаков с помощью управляемой значимости (backprop_modifier='guided') и устранения деконверсии (backprop_modifier='relu').

2. gradCAM - ванильный, управляемый, ректифицированный. Они должны содержать больше деталей, поскольку в них используются объекты Conv или объединения, которые содержат больше пространственных деталей, которые теряются в плотных слоях. Единственной дополнительной деталью по сравнению с заметностью является penultimate_layer_idx. Здесь указывается предварительный слой, градиенты которого следует использовать. Технические подробности смотрите в этой статье: https://arxiv.org/pdf/1610.02391v1.pdf. По умолчанию, если значение penultimate_layer_idx не определено, выполняется поиск ближайшего предварительного слоя. Для нашей архитектуры это был бы уровень block

5_pool после всех слоев Conv. Получить краткое описание модели с помощью команды model.summary().

a. Выполнить построение карт значимости классов для выбранных изображений методами ванильный, управляемый, ректифицированный.

b. Сделать выводы о наиболее точном и полном методе описания активаций слоев нейронной сети.

Установим у tf-keras-vis:

```
!pip install tf-keras-vis
```

Executed at 2024.01.20 14:11:16 in 3s 755ms

```
Collecting tf-keras-vis
  Downloading tf_keras_vis-0.8.6-py3-none-any.whl (52 kB)
     ---------------------------------------- 52.1/52.1 kB 1.3 MB/s eta 0:00:00
Requirement already satisfied: pillow in c:\python\aszii\lib\site-packages (from tf-keras-vis) (10.1.0)
Requirement already satisfied: packaging in c:\python\aszii\lib\site-packages (from tf-keras-vis) (23.2)
Requirement already satisfied: imageio in c:\python\aszii\lib\site-packages (from tf-keras-vis) (2.33.1)
Collecting deprecated
  Downloading Deprecated-1.2.14-py2.py3-none-any.whl (9.6 kB)
Requirement already satisfied: scipy in c:\python\aszii\lib\site-packages (from tf-keras-vis) (1.11.4)
Requirement already satisfied: wrapt<2,>=1.10 in c:\python\aszii\lib\site-packages (from deprecated->tf-keras-vis) (1.14.1)
Requirement already satisfied: numpy in c:\python\aszii\lib\site-packages (from imageio->tf-keras-vis) (1.26.1)
Installing collected packages: deprecated, tf-keras-vis
Successfully installed deprecated-1.2.14 tf-keras-vis-0.8.6

WARNING: Error parsing requirements for torch: [Errno 2] No such file or directory: 'c:\\python\\aszii\\lib\\site-packages\\torch-2.1.0.dist-info\\METADATA'

[notice] A new release of pip available: 22.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Загрузим предобученную модель VGG16 и выберем 4 изображения, отобразив их.

```
model = Model(weights='imagenet', include_top=True)

imgTitleList = ['goldfish', 'great_white_shark', 'tiger_shark', 'hen']

imgPathList = ['n01443537_goldfish.JPEG', 'n01484850_great_white_shark.JPEG', 'n01491361_tiger_shark.JPEG', 'n01514859_hen.JPEG']

imgArr = np.asarray([np.array(load_img(imgPath, target_size=(224, 224))) for imgPath in imgPathList])

X = preprocess_input(imgArr)

f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))
for i, title in enumerate(imgTitleList):
    ax[i].set_title(title, fontsize=16)
    ax[i].imshow(imgArr[i])
    ax[i].axis('off')
plt.tight_layout()
plt.show()
```

Executed at 2024.01.20 14:45:07 in 3s 369ms

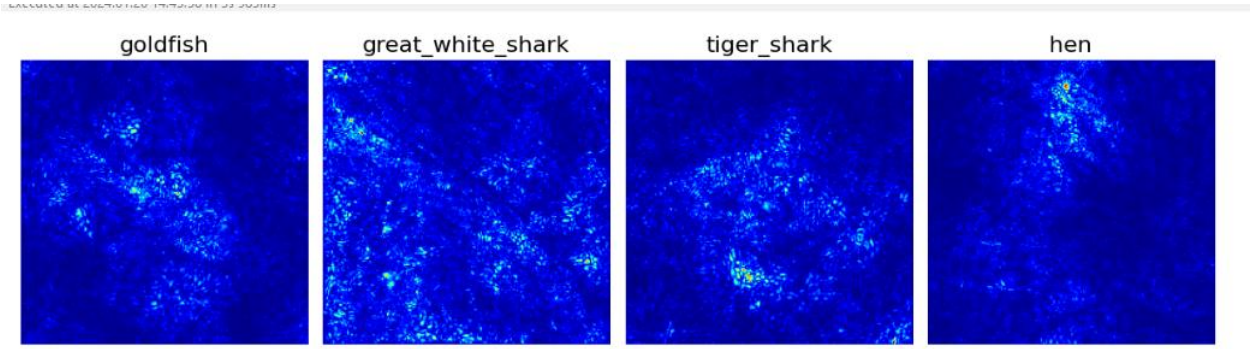
Заменим функцию активации на линейную и установим классы изображения.

```
replace2linear = ReplaceToLinear()
def model_modifier_function(cloned_model):
    cloned_model.layers[-1].activation = tf.keras.activations.linear

score = CategoricalScore([1, 2, 3, 8])
def score_function(output):
    return (output[0][1], output[1][2], output[2][3], output[3][8])
```
Executed at 2024.01.20 14:45:47 in 57ms

Отобразим карты значимости с помощью

Saliency vanila: Этот метод вычисляет градиент выхода модели по отношению к входу и использует абсолютное значение градиента для оценки важности каждого пикселя входного изображения.



Так как visualize_saliency не заработал использовал SmoothGrad

Этот метод добавляет случайный шум к входу и усредняет карты важности, получаемые для нескольких шумовых вариантов входного изображения. Это помогает устранить шум и сгладить карту важности.

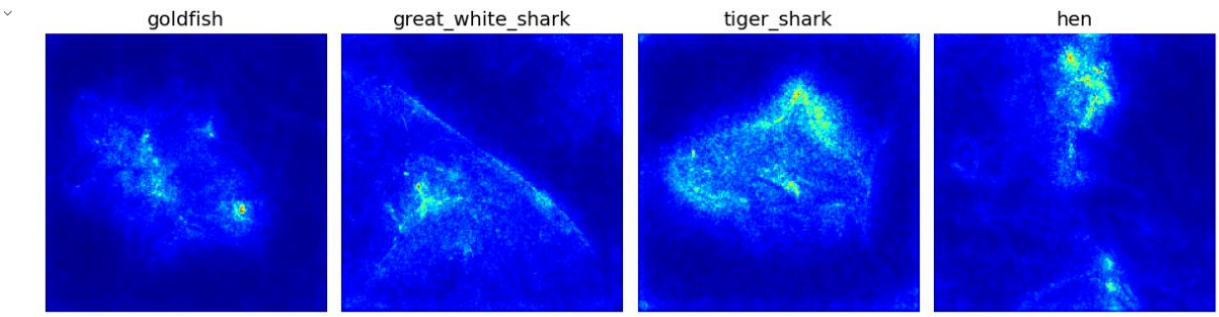
SmoothGrad

```
mapList = 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(imgTitleList):
    ax[i].set_title(title, fontsize=14)
    ax[i].imshow(mapList[i], cmap='jet')
    ax[i].axis('off')

plt.tight_layout()
plt.show()
```

Executed at 2024.01.20 17:49:07 in 59s 10ms



GradCAM

GradCAM (Gradient-weighted Class Activation Mapping) - это метод, используемый для генерации карты активации класса на основе градиентов выхода модели по отношению к определенному классу. Это позволяет понять, какие области изображения наиболее сильно влияют на классификацию модели.

GradCAM использует градиентные значения из последнего сверточного слоя модели для взвешивания их важности и создания тепловой карты, которая показывает области изображения, активирующие данный класс. Это позволяет интерпретировать решения модели и понять, где происходят детекции объектов или где находятся существенные признаки, влияющие на классификацию.

Визуализируем карты активации через GradCAM

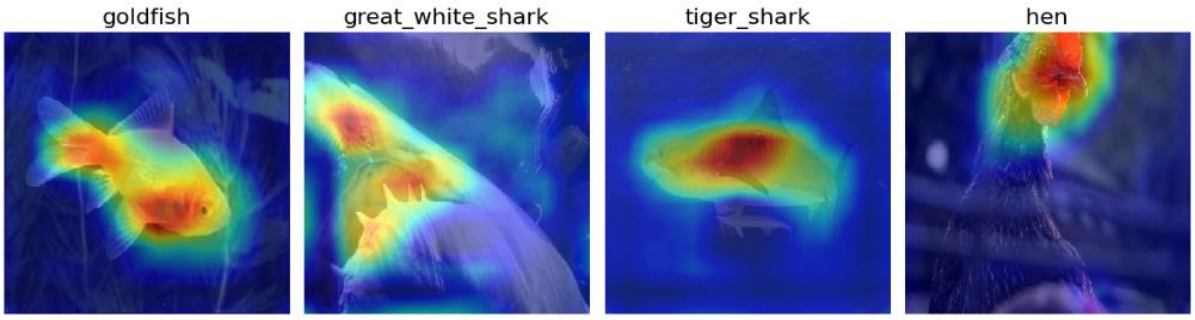
GrandCAM

```
gradcam = Gradcam(model,model_modifier=replace2linear,clone=True)
mapList = gradcam(score,X,penultimate_layer=-1)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))

for i, title in enumerate(imgTitleList):
    heatmap = np.uint8(cm.jet(mapList[i])[..., :4] * 255)
    ax[i].set_title(title, fontsize=16)
    ax[i].imshow(imgArr[i])
    ax[i].imshow(heatmap, cmap='jet', alpha=0.5)
    ax[i].axis('off')

plt.tight_layout()
plt.show()
```

Executed at 2024.01.20 14:47:17 in 5s 635ms



Также воспользуемся GradCAM++

GradCAM++ (Gradient-weighted Class Activation Mapping++) – это улучшенная версия метода GradCAM, предназначенная для более точной генерации карт активации класса в моделях глубокого обучения. GradCAM++ вносит изменения в оригинальный алгоритм, чтобы лучше учитывать контекст информации изображения.

В отличие от GradCAM, который фокусируется только на градиентах, GradCAM++ использует еще один важный элемент – информацию о холмовой функции, выходе модели, которая обозначает активацию класса. Эта дополнительная информация позволяет более точно определить области изображения, которые вносят наибольший вклад в классификацию.

Визуализируем

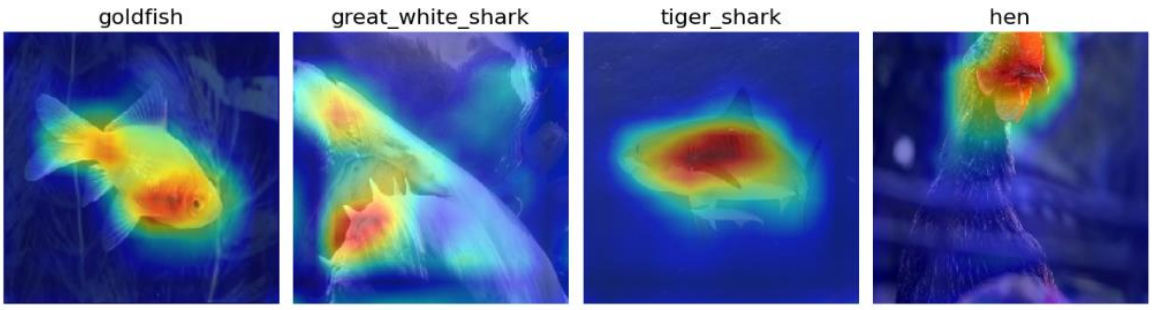
GrandCAM++

```
gradcam = GradcamPlusPlus(model,model_modifier=replace2linear,clone=True)
mapList = gradcam(score,X,penultimate_layer=-1)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))

for i, title in enumerate(imgTitleList):
    heatmap = np.uint8(cm.jet(mapList[i])[..., :4] * 255)
    ax[i].set_title(title, fontsize=16)
    ax[i].imshow(imgArr[i])
    ax[i].imshow(heatmap, cmap='jet', alpha=0.5)
    ax[i].axis('off')

plt.tight_layout()
plt.show()
```

Executed at 2024.01.20 14:47:22 in 5s 485ms



Результирующий вывод:

SmoothGrad, GradCAM и GradCAM++ являются тремя различными методами для генерации карт активации класса (saliency maps) и визуализации важных областей изображений в моделях глубокого обучения.

1. SmoothGrad:

SmoothGrad - это метод, который добавляет случайный шум к входу и усредняет карты важности, полученные для нескольких шумовых вариантов входного изображения. Это помогает сгладить карту важности, снизить шум и улучшить качество визуализации. SmoothGrad имеет преимущество в устранении некоторых шумовых артефактов, которые могут появиться в других методах.

2. GradCAM:

GradCAM (Gradient-weighted Class Activation Mapping) - это метод, который использует градиенты из последнего сверточного слоя модели для расчета важности каждого пикселя входного изображения. Чем выше градиент, тем больше пиксель влияет на классификацию. GradCAM

позволяет понять, какие области изображения больше всего влияют на предсказания модели, и визуализировать эти области.

3. GradCAM++:

GradCAM++ (Gradient-weighted Class Activation Mapping++) - это улучшенная версия GradCAM, которая также учитывает информацию о холмовой функции, выходе модели, помимо градиентов. Информация о холмовой функции помогает более точно определить области изображения, которые вносят наибольший вклад в классификацию. GradCAM++ предоставляет более детализированные и точные карты активации класса, обеспечивая лучшую интерпретируемость решений модели.

Отличия между этими методами заключаются в подходе к вычислению важности пикселей и методах сглаживания или учета дополнительных данных, таких как градиенты или информация о холмовой функции. Каждый метод имеет свои преимущества и может быть применен в зависимости от конкретных задач и требований.