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

ИКБ направление «Кибербразведка 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 систем искусственного интеллекта»

Группа:
ББМО-02-22
Выполнил:
Шитов А.В.

Проверил:
к.т.н Спирин А.А.

Москва 2023

Использование механизмов внимания в нейронных сетях

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

a. Использовать репозиторий

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',

d. Отобразить карты значимости признаков, найти и задать верный

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: 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.

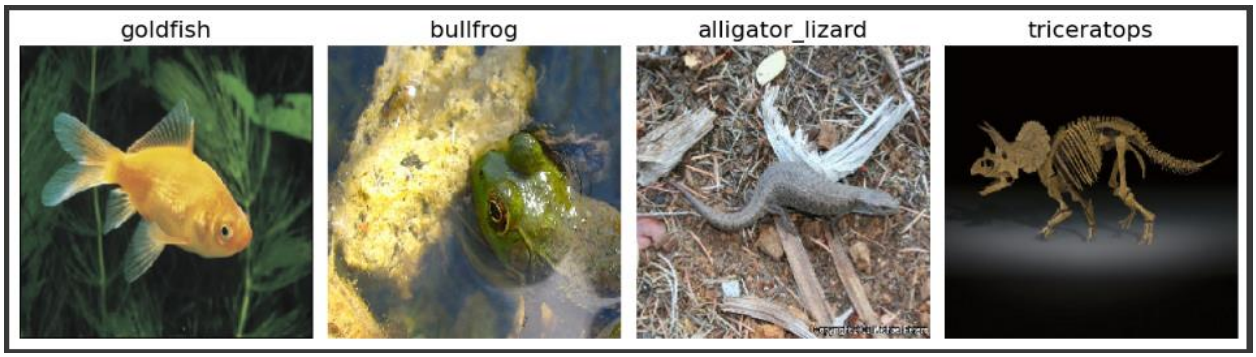
Установка tf-keras-vis

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

     Requirement already satisfied: tf-keras-vis in c:\python\aszii\lib\site-packages (0.8.6)
     Requirement already satisfied: scipy in c:\python\aszii\lib\site-packages (from tf-keras-vis) (1.11.4)
     Requirement already satisfied: pillow in c:\python\aszii\lib\site-packages (from tf-keras-vis) (10.1.0)
     Requirement already satisfied: deprecated in c:\python\aszii\lib\site-packages (from tf-keras-vis) (1.2.14)
     Requirement already satisfied: imageio in c:\python\aszii\lib\site-packages (from tf-keras-vis) (2.33.1)
     Requirement already satisfied: packaging in c:\python\aszii\lib\site-packages (from tf-keras-vis) (23.2)
     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)
```

Загрузка предобученной модели VGG16

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

     imgTitleList = ['goldfish', 'bullfrog', 'alligator_lizard', 'triceratops']

     imgPathList = ['n01443537_goldfish.JPEG', 'n01641577_bullfrog.JPEG', 'n01689811_alligator_lizard.JPEG', 'n01704323_triceratops.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()
```

Вывод 4 выбранных изображений

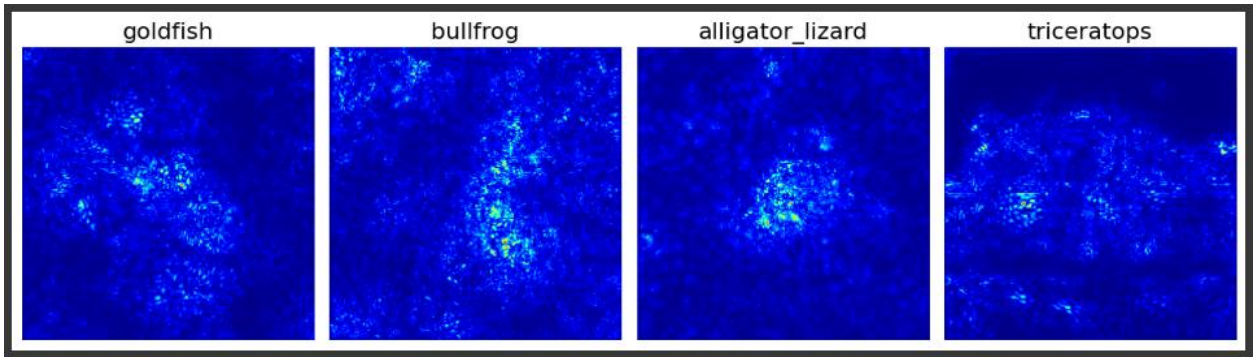


Далее производится замена функции активации на линейную и установка классов изображений.

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

     score = CategoricalScore([1, 30, 44, 51])
     def score_function(output):
         return (output[0][1], output[1][30], output[2][44], output[3][51])
```

Отображение карт значимости

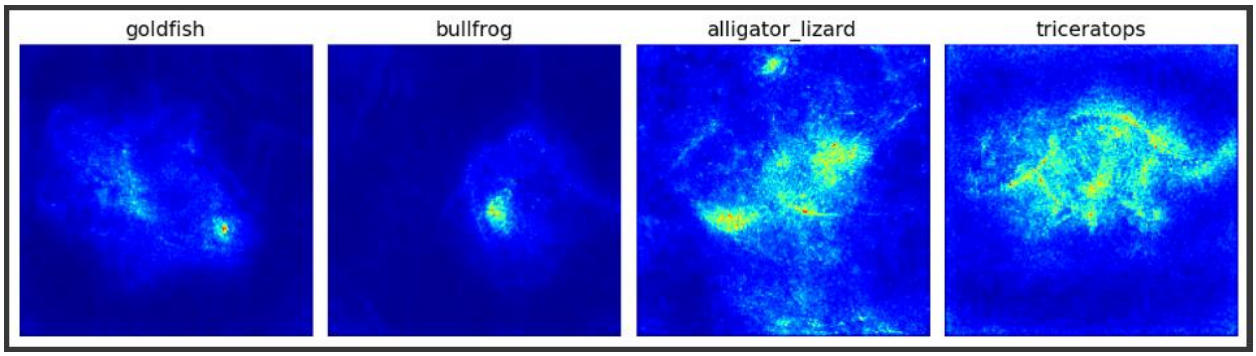
Вычисляет градиент выхода модели по отношению к входу и использует абсолютное значение градиента для оценки важности каждого пикселя входного изображения.



Использование ранее указанного visualize_saliency не привело к нужному результату, поэтому было принято решение воспользоваться SmoothGrad. Данный метод добавляет случайный шум к входу и усредняет карты важности, получаемые для нескольких шумовых вариантов входного изображения, что позволяет 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()
```



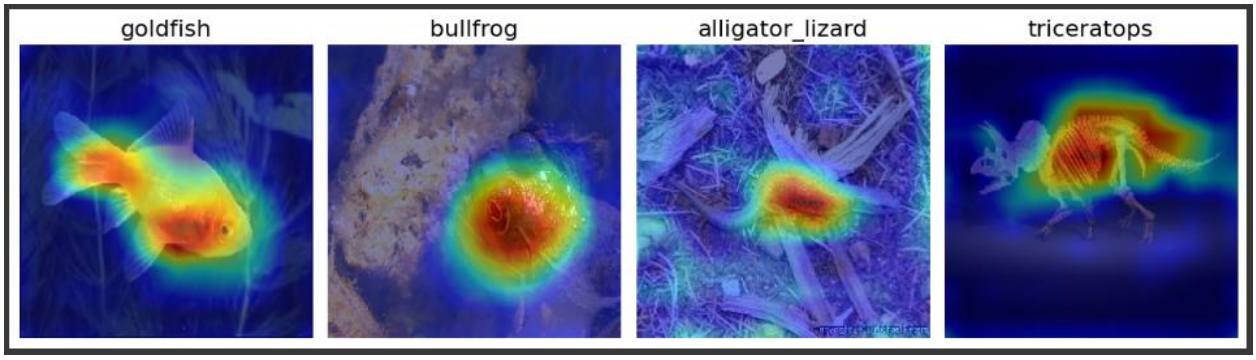
Gradient-weighted Class Activation Mapping) - это метод визуализации важности пикселей или областей изображения для принятия решения нейросетью. Он позволяет понять, какие части изображения привлекают внимание нейросети при классификации или детектировании объектов.

GradCAM базируется на анализе градиентов активации по отношению к конкретному классу. Он обеспечивает локализацию объектов и позволяет определить влияние каждого пикселя на принятие решения нейросетью.

Визуализация карты активации при помощи GradCAM отображена ниже.

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()
```

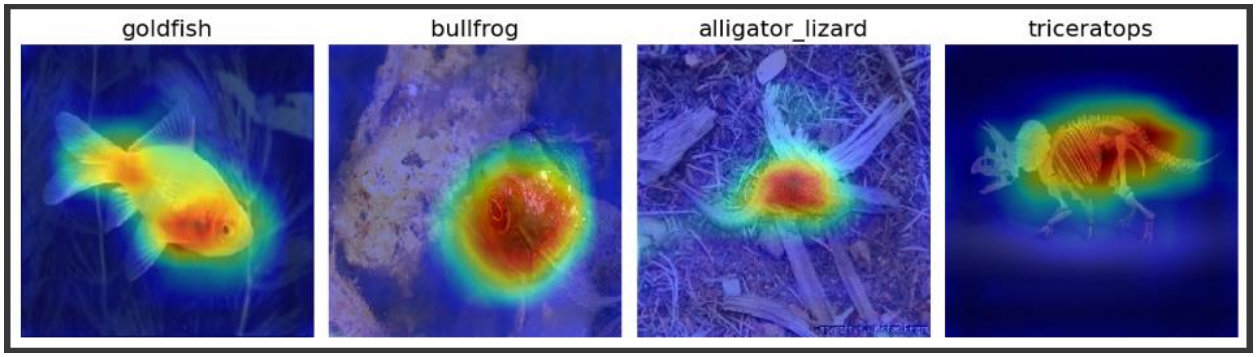


d Class Activation Mapping++) является улучшенной версией метода GradCAM, которая предназначена для более точной генерации карт активации класса в моделях глубокого обучения.

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.

Визуализация отображена на рисунках ниже.

```
[ ]  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()
```



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.

SmoothGrad является методом добавления случайного шума к изображению и усреднения результатов классификации для различных случайных вариантов.

Этот метод помогает устранить случайное влияние шума и сгладить градиенты активации, позволяя более точный анализ важности пикселей.

SmoothGrad полезен для локализации ключевых объектов, обнаружения особых областей и оценки надежности предсказаний.

позволяет визуализировать важность различных областей изображения для принятия решения нейросетью.

Этот метод вычисляет градиенты активации по отношению к конкретному классу и затем использует их для взвешивания значимости пикселей.

предоставляет более точную локализацию объектов и позволяет понять, какие области в изображении привлекают внимание нейросети при классификации.

GradCAM++ является расширением GradCAM с уточненным определением важности пикселей. Добавленные компоненты в GradCAM++

позволяют учесть более сложные зависимости между активациями и

г

р Этот метод улучшает точность и позволяет более точно определить

области интереса в изображении.

д

Итог

и

е

Отличия между этими методами заключаются в подходе к вычислению

н

важности пикселей и методах сглаживания или учета дополнительных данных,

т

таких как градиенты или информация о холмовой функции.

а

Выбор метода зависит от конкретных задач и требований. GradCAM и

м

GradCAM++ обеспечивают более точные результаты и более четкую

и

локализацию объектов, но требуют больше вычислительных ресурсов.

SmoothGrad является более простым методом, но имеет свои преимущества.

в

н

е

й

р

о

с

е

т

и