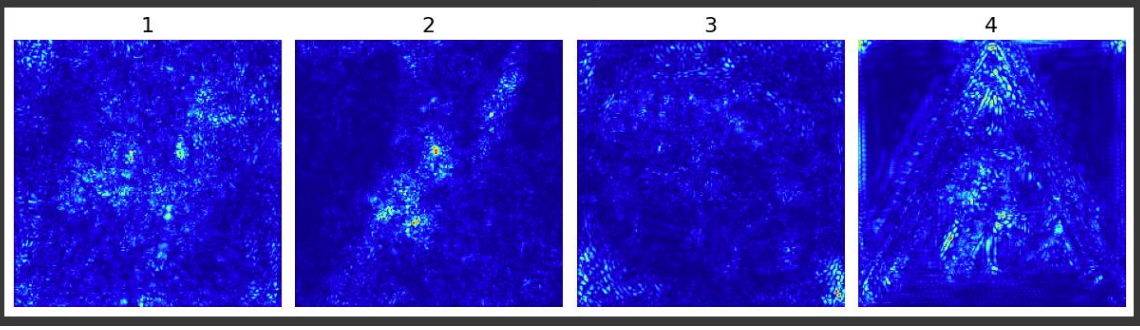
Загрузим 4 изображения и отобразим их

```python
# Загружаем предобученную модель с весами ImageNet и включаем верхний слой (классификация)
model = Model(weights='imagenet', include_top=True)

# Список заголовков для изображений
imgTitleList = ['1', '2', '3', '4']

# Список путей к изображениям
imgPathList = ['1.jpg', '2.jpg', '3.jpg', '4.jpg']

# Загружаем изображения, изменяем их размер до 224x224 пикселей и преобразуем в массив NumPy
imgArr = np.asarray([np.array(load_img(imgPath, target_size=(224, 224))) for imgPath in imgPathList])

# Предобрабатываем изображения для подачи в модель (нормализация и изменение формата)
X = preprocess_input(imgArr)

# Создаем фигуру и оси для отображения изображений (1 строка, 4 колонки)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))

# Цикл по заголовкам изображений и индексам
for i, title in enumerate(imgTitleList):
    ax[i].set_title(title, fontsize=16)  # Устанавливаем заголовок для каждого изображения
    ax[i].imshow(imgArr[i])  # Отображаем изображение на соответствующей оси
    ax[i].axis('off')  # Отключаем отображение осей (графиков) для более чистого вида

# Настраиваем компоновку графиков
plt.tight_layout()

# Показываем финальный график с изображениями
plt.show()
```

Отобразим карты значимости

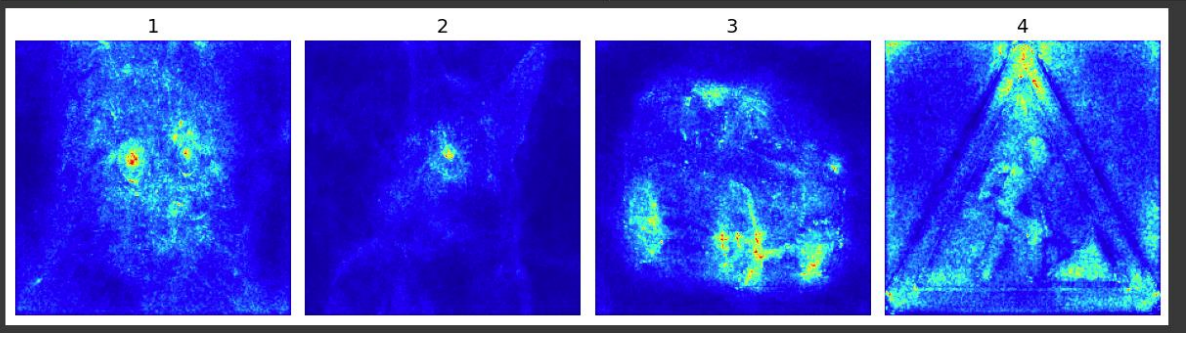
```python
[7] # Создаем объект Saliency для вычисления карты значимости
saliency = Saliency(model, model_modifier=replace2linear, clone=True)

# Вычисляем карты значимости для заданных входных данных X с использованием заданного объекта score
mapList = saliency(score, X)

# Создаем фигуру и оси для отображения карт значимости (1 строка, 4 колонки)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))

# Цикл по заголовкам изображений и индексам
for i, title in enumerate(imgTitleList):
    ax[i].set_title(title, fontsize=16)  # Устанавливаем заголовок для каждой карты значимости
    ax[i].imshow(mapList[i], cmap='jet')  # Отображаем карту значимости с использованием цветовой карты 'jet'
    ax[i].axis('off')  # Отключаем отображение осей (графиков) для более чистого вида

# Настраиваем компоновку графиков
plt.tight_layout()

# Показываем финальный график с картами значимости
plt.show()
```



SmoothGrad

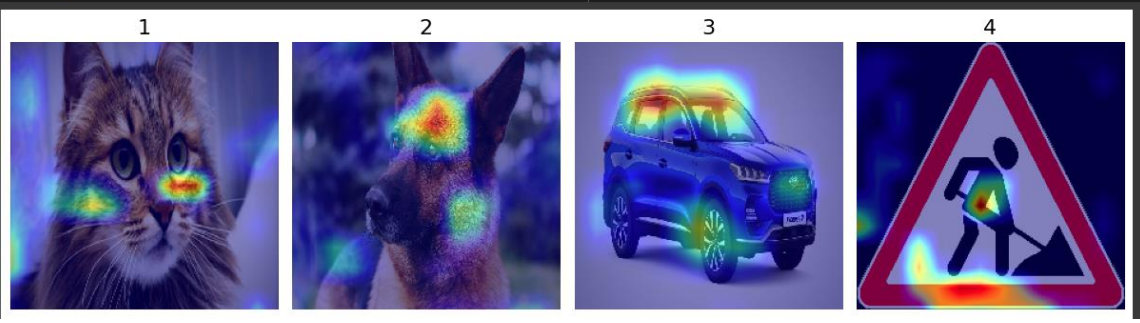
SmoothGrad

```
# Вычисляем карты значимости с использованием метода saliency, добавляя параметры для сглаживания
mapList = saliency(score, X, smooth_samples=20, smooth_noise=0.20)

# Создаем фигуру и оси для отображения карт значимости (1 строка, 4 колонки)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))

# Цикл по заголовкам изображений и индексам
for i, title in enumerate(imgTitleList):
    ax[i].set_title(title, fontsize=14)  # Устанавливаем заголовок для каждой карты значимости
    ax[i].imshow(mapList[i], cmap='jet')  # Отображаем карту значимости с использованием цветовой карты 'jet'
    ax[i].axis('off')  # Отключаем отображение осей (графиков) для более чистого вида

# Настраиваем компоновку графиков
plt.tight_layout()

# Показываем финальный график с картами значимости
plt.show()
```



GrandCAM

```
# Создаем объект Gradcam для вычисления карт Grad-CAM
gradcam = Gradcam(model, model_modifier=replace2linear, clone=True)

# Вычисляем карты Grad-CAM для заданных входных данных X, указывая предпоследний слой
mapList = gradcam(score, X, penultimate_layer=-1)

# Создаем фигуру и оси для отображения изображений с картами Grad-CAM (1 строка, 4 колонки)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))

# Цикл по заголовкам изображений и индексам
for i, title in enumerate(imgTitleList):
    # Преобразуем карту значимости в формат изображения (целочисленный) с использованием цветовой карты 'jet'
    heatmap = np.uint8(cm.jet(mapList[i])[..., :4] * 255)  # Применяем цветовую карту и масштабируем значения

    ax[i].set_title(title, fontsize=16)  # Устанавливаем заголовок для каждой карты Grad-CAM
    ax[i].imshow(imgArr[i])  # Отображаем оригинальное изображение
    ax[i].imshow(heatmap, cmap='jet', alpha=0.5)  # Накладываем тепловую карту на изображение с прозрачностью 0.5
    ax[i].axis('off')  # Отключаем отображение осей (графиков) для более чистого вида

# Настраиваем компоновку графиков
plt.tight_layout()

# Показываем финальный график с изображениями и картами Grad-CAM
plt.show()
```



GrandCAM++

```python
# Создаем объект GradcamPlusPlus для вычисления карт Grad-CAM++
gradcam = GradcamPlusPlus(model, model_modifier=replace2linear, clone=True)

# Вычисляем карты Grad-CAM++ для заданных входных данных X, указывая предпоследний слой
mapList = gradcam(score, X, penultimate_layer=-1)

# Создаем фигуру и оси для отображения изображений с картами Grad-CAM++ (1 строка, 4 колонки)
f, ax = plt.subplots(nrows=1, ncols=4, figsize=(12, 4))

# Цикл по заголовкам изображений и индексам
for i, title in enumerate(imgTitleList):
    # Преобразуем карту значимости в формат изображения (целочисленный) с использованием цветовой карты 'jet'
    heatmap = np.uint8(cm.jet(mapList[i])[..., :4] * 255)  # Применяем цветовую карту и масштабируем значения

    ax[i].set_title(title, fontsize=16)  # Устанавливаем заголовок для каждой карты Grad-CAM++
    ax[i].imshow(imgArr[i])  # Отображаем оригинальное изображение
    ax[i].imshow(heatmap, cmap='jet', alpha=0.5)  # Накладываем тепловую карту на изображение с прозрачностью 0.5
    ax[i].axis('off')  # Отключаем отображение осей (графиков) для более чистого вида

# Настраиваем компоновку графиков
plt.tight_layout()

# Показываем финальный график с изображениями и картами Grad-CAM++
plt.show()
```