TOPIC **: "Explainable AI using class activation mappings in Pytorch"**

GROUP NUMBER **: 7**

TEAM MEMBERS **: Kashif Riyaz, Nitesh Morem**

UNDER THE GUIDANCE **: Prof. Dr.-Ing. Christian Bergler**

We Used the Jacob gills github Respository while working on the code.

The Repository Link is"https://github.com/jacobgil/pytorch-grad-cam"

0 Installing necessary Modules

In this sections all the necessary modules are installed which include:

- Pytorch
- torchvision
- numpy
- opencv
- matplotlib
- grad-cam

Note: The Grad-CAM modules have been imported from the jacob gill's github repository: reference link: https://github.com/jacobgil/pytorch-grad-cam

In [91]:
```
!pip install --upgrade pip
!pip install torch torchvision numpy opencv-python requests pillow grad-cam matp
```

```
Requirement already satisfied: pip in c:\users\kashi\virtualenv\ai_projects\lib\s
ite-packages (24.0)
Collecting pip
  Using cached pip-24.1.1-py3-none-any.whl.metadata (3.6 kB)
Using cached pip-24.1.1-py3-none-any.whl (1.8 MB)
ERROR: To modify pip, please run the following command:
C:\Users\kashi\virtualenv\AI_Projects\Scripts\python.exe -m pip install --upgrade
pip

[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: torch in c:\users\kashi\virtualenv\ai_projects\lib
\site-packages (2.2.2)
Requirement already satisfied: torchvision in c:\users\kashi\virtualenv\ai_projec
ts\lib\site-packages (0.17.2)
Requirement already satisfied: numpy in c:\users\kashi\virtualenv\ai_projects\lib
\site-packages (1.26.4)
Requirement already satisfied: opencv-python in c:\users\kashi\virtualenv\ai_proj
ects\lib\site-packages (4.9.0.80)
Requirement already satisfied: requests in c:\users\kashi\virtualenv\ai_projects
\lib\site-packages (2.31.0)
Requirement already satisfied: pillow in c:\users\kashi\virtualenv\ai_projects\li
b\site-packages (10.3.0)
Requirement already satisfied: grad-cam in c:\users\kashi\virtualenv\ai_projects
\lib\site-packages (1.5.0)
Requirement already satisfied: matplotlib in c:\users\kashi\virtualenv\ai_project
s\lib\site-packages (3.8.4)
Requirement already satisfied: filelock in c:\users\kashi\virtualenv\ai_projects
\lib\site-packages (from torch) (3.13.4)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\kashi\virtual
env\ai_projects\lib\site-packages (from torch) (4.11.0)
Requirement already satisfied: sympy in c:\users\kashi\virtualenv\ai_projects\lib
\site-packages (from torch) (1.12)
Requirement already satisfied: networkx in c:\users\kashi\virtualenv\ai_projects
\lib\site-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in c:\users\kashi\virtualenv\ai_projects\li
b\site-packages (from torch) (3.1.3)
Requirement already satisfied: fsspec in c:\users\kashi\virtualenv\ai_projects\li
b\site-packages (from torch) (2024.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\kashi\virtual
env\ai_projects\lib\site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\kashi\virtualenv\ai_proje
cts\lib\site-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\kashi\virtualenv\ai
_projects\lib\site-packages (from requests) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\kashi\virtualenv\ai
_projects\lib\site-packages (from requests) (2024.2.2)
Requirement already satisfied: ttach in c:\users\kashi\virtualenv\ai_projects\lib
\site-packages (from grad-cam) (0.0.3)
Requirement already satisfied: tqdm in c:\users\kashi\virtualenv\ai_projects\lib
\site-packages (from grad-cam) (4.66.2)
Requirement already satisfied: scikit-learn in c:\users\kashi\virtualenv\ai_proje
cts\lib\site-packages (from grad-cam) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\kashi\virtualenv\ai_p
rojects\lib\site-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\kashi\virtualenv\ai_proje
cts\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\kashi\virtualenv\ai_
projects\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\kashi\virtualenv\ai_
projects\lib\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\kashi\virtualenv\ai_pr
ojects\lib\site-packages (from matplotlib) (24.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\kashi\virtualenv\ai_p
rojects\lib\site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kashi\virtualenv
\ai_projects\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\kashi\virtualenv\ai_projects
\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\kashi\virtualenv\ai_pr
ojects\lib\site-packages (from jinja2->torch) (2.1.5)
```

```
Requirement already satisfied: scipy>=1.6.0 in c:\users\kashi\virtualenv\ai_proje
cts\lib\site-packages (from scikit-learn->grad-cam) (1.13.0)
Requirement already satisfied: joblib>=1.2.0 in c:\users\kashi\virtualenv\ai_proj
ects\lib\site-packages (from scikit-learn->grad-cam) (1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\kashi\virtualenv
\ai_projects\lib\site-packages (from scikit-learn->grad-cam) (3.4.0)
Requirement already satisfied: mpmath>=0.19 in c:\users\kashi\virtualenv\ai_proje
cts\lib\site-packages (from sympy->torch) (1.3.0)
Requirement already satisfied: colorama in c:\users\kashi\virtualenv\ai_projects
\lib\site-packages (from tqdm->grad-cam) (0.4.6)
[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

1. Importing necessary modules

In this sections all the modules are imported

In [92]:
```
import torchvision
import warnings
import torch
warnings.filterwarnings('ignore')
from torchvision import models
import numpy as np
import cv2
import requests
from PIL import Image

import matplotlib.pyplot as plt
```

Note: if there is an error regarding any module please restart the kernal and run the whole code again

1.1 Importing modules from pytorch_grad_cam

In [93]:
```
from pytorch_grad_cam.utils.model_targets import ClassifierOutputTarget
from pytorch_grad_cam.utils.image import show_cam_on_image, \
    deprocess_image, \
    preprocess_image
```

1.1.1 Importing modules for visualisation

In [94]:
```
from pytorch_grad_cam.grad_cam_plusplus import GradCAMPlusPlus
from pytorch_grad_cam.eigen_cam import EigenCAM
from pytorch_grad_cam.score_cam import ScoreCAM
from pytorch_grad_cam.xgrad_cam import XGradCAM
from pytorch_grad_cam import GradCAM
```

2. Defining necessary functions

In this section all fucntions which are necessary for this notebook are defined which include:

- image_transform()
- predict_and_get_target_class()
- get_conv_layer()

In [95]:
```python
def image_transform(image_url):
    """
    Transforms an image from a given URL into a preprocessed image tensor suitab

    Parameters:
    - image_url (str): The URL of the image to be transformed.

    Returns:
    - img (numpy.ndarray): The resized image as a numpy array, normalized to the
    - input_tensor (torch.Tensor): The preprocessed image tensor ready for model

    Process:
    1. The image is loaded from the provided URL.
    2. The image is resized to 224x224 pixels.
    3. The pixel values are scaled to the range [0, 1].
    4. The image is normalized using the specified mean and standard deviation.
    """
    img = np.array(Image.open(requests.get(image_url, stream=True).raw))
    img = cv2.resize(img, (224, 224))
    img = np.float32(img) / 255
    input_tensor = preprocess_image(img, mean=[0.485, 0.456, 0.406], std=[0.229,
    return img, input_tensor
```

In [96]:
```python
def predict_and_get_target_class(model, input_tensor):
    """
    Predicts the class of the given input tensor using the provided model and re

    Parameters:
    - model (torch.nn.Module): The neural network model used for making predicti
    - input_tensor (torch.Tensor): The preprocessed image tensor from image_tran

    Returns:
    - predicted_class (list): A list containing the predicted class index.
    - predicted_label (list): A list containing the predicted label name.

    Process:
    1. Passes the input tensor through the model to get the output.
    2. Determines the index of the highest value in the output tensor (using arg
    3. Returns the predicted class index as a list (list of one value).
    4. Maps the predicted class index to the corresponding human-readable label
    """
    # Fetch the ImageNet labels
    response = requests.get('https://raw.githubusercontent.com/pytorch/hub/maste
    labels = response.text.strip().split('\n')

    # Ensure the model is in evaluation mode
    model.eval()

    with torch.no_grad():
        # Get the model's output for the input tensor
        output = model(input_tensor)

        # Get the index of the highest value in the output tensor
        predicted_class = [torch.argmax(output).item()]
```

```
        # Get the human-readable label name using the predicted class index
        predicted_label = labels[predicted_class[0]]

    return predicted_class, predicted_label
```

In [97]:
```
def get_conv_layer(model):
    """
    Retrieves a specified convolutional layer from the given neural network mode

    Parameters:
    - model (torch.nn.Module): The neural network model to search through.

    Returns:
    - torch.nn.Conv2d: The selected convolutional layer based on user input.

    Process:
    1. Collects all convolutional layers (`torch.nn.Conv2d`) from the model.
    2. Prints the total number of convolutional layers found.
    3. Prompts the user to input the index of the convolutional layer they wish
    4. Checks if the user input is a valid layer number.
    5. Returns the convolutional layer.
    6. If the input is not valid, prints an error message.

    Note:
    - If there are no convolutional layers in the model, the function will not p
    """
    conv_layers = [layer for layer in model.modules() if isinstance(layer, torch
    input_layer = int(input(f'There are {len(conv_layers)} conv2d layers in this
    if input_layer <= 0 :
        return f'Please enter positive layer number'

    elif input_layer > len(conv_layers):
        return f'Please enter proper layer number'
    else:
        return conv_layers[input_layer - 1]
```

3. Importing image and model

In this section we use the above functions and also load the model and input image

In [98]:
```
# storing the model in a variable.
model = models.densenet121(pretrained=True)
#model.eval()
```

In [99]:
```
# storing the URL of the image in a variable.
image_url='https://media.istockphoto.com/id/492696926/photo/wood-mouse-on-root-o
img,input_tensor=image_transform(image_url)
```

3.1.1 Making prediction with the model

In [100…
```
# making prediction on the input tensor based on our model.
index,name=predict_and_get_target_class(model,input_tensor)
index,name
```

Out[100… ([331], 'hare')

3.1.2 Saving the target label number

In [101…
```
# saving the prediction on the variable targets.(optional for Grad-CAM)
targets =[ClassifierOutputTarget(index)]
```

3.1.3 Getting the convolutional layer

In [102…
```
# Getting the target layer of the model for visualization.
target_layers = get_conv_layer(model)
target_layers
```

Out[102… Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

4. Appling the Gradcam methods

In this section all the CAM techniques are applied on model and input tensor which include:

- Grad-CAM
- Grad-CAM++
- Eigen-CAM
- XGrad-CAM

Note: These techniques are taken from jacob gill's github repository reference link: https://github.com/jacobgil/pytorch-grad-cam

4.1 Normal Grad-CAM

In [103…
```
with GradCAM(model=model, target_layers=[target_layers]) as cam:
    grayscale_cams = cam(input_tensor=input_tensor, targets=targets)
    grad_cam_image = show_cam_on_image(img, grayscale_cams[0, :], use_rgb=True)
```

4.2 GradCAM++

In [104…
```
cam2=GradCAMPlusPlus(model=model, target_layers=[target_layers])
grayscale_cams2=cam2(input_tensor=input_tensor, targets=targets)
grayscale_cam2 = grayscale_cams2[0, :]
gradcamplusplus_image=show_cam_on_image(img,grayscale_cam2,use_rgb=True)
```

4.3 EigenCAM

In [105…
```
cam3=EigenCAM(model=model,target_layers=[target_layers])
grayscale_cams3=cam3(input_tensor=input_tensor)
grayscale_cam3=grayscale_cams3[0, :]
eigencam_image=show_cam_on_image(img,grayscale_cam3,use_rgb=True)
```

4.4 ScoreCAM

Here we have not shown the implementation of ScoreCAM because it takes a lot of time and the output is also not included in the results:

In [106…
```
'''cam4=ScoreCAM(model=model,target_layers=target_layers)
grayscale_cams4=cam4(input_tensor=input_tensor)
grayscale_cam4=grayscale_cams4[0, :]
cam4_image=show_cam_on_image(img,grayscale_cam4,use_rgb=True)'''
```

Out[106…
```
'cam4=ScoreCAM(model=model,target_layers=target_layers)\ngrayscale_cams4=cam4(i
nput_tensor=input_tensor)\ngrayscale_cam4=grayscale_cams4[0, :]\ncam4_image=sho
w_cam_on_image(img,grayscale_cam4,use_rgb=True)'
```

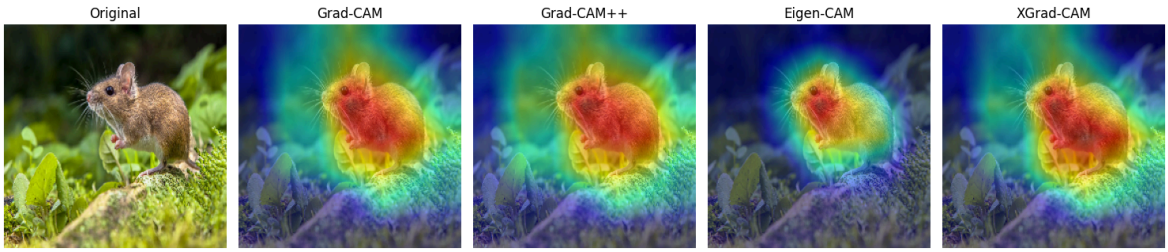
4.5 XGradCAM

In [107…
```
cam5=XGradCAM(model=model,target_layers=[target_layers])
grayscale_cams5=cam5(input_tensor=input_tensor)
grayscale_cam5=grayscale_cams5[0, :]
xgradcam_image=show_cam_on_image(img,grayscale_cam5,use_rgb=True)
```

5. visualizing

In this section the visulization takes place

In [108…
```
# here we plot the visualizations and  aslo print the label name of the  based o
cam = np.uint8(255*grayscale_cams[0, :])
cam = cv2.merge([cam, cam, cam])
labels = ['Original', 'Grad-CAM', 'Grad-CAM++', 'Eigen-CAM', 'XGrad-CAM']
imagess = np.hstack((np.uint8(255*img), grad_cam_image, gradcamplusplus_image,ei
fig, axes = plt.subplots(1, len(labels), figsize=(15, 5))
for ax, image, label in zip(axes, [img, grad_cam_image, gradcamplusplus_image, e
    ax.imshow(image)
    ax.set_title(label)
    ax.axis('off')
plt.tight_layout()
plt.show()
print(f'The above image has {name} in it.')
```




```
The above image has hare in it.
```

The above picture highlights the parts of the image that the model focused on to make that prediction.

The highlighted part varies slightly between different techniques because each method uses distinct approaches to emphasize the regions.