CSCI323 Lab 1 Assignment (2025)

Content modified by: Cher Lim (cherl@uow.edu.au)

Let's also create a direct comparison between the two models

```
# Let's also create a direct comparison between the two models
def compare_saliency_maps():
    """
    Generate and compare saliency maps between the two models
    """
    # Get a batch of test images
    dataiter = iter(testloader)
    images, labels = next(dataiter)

    # Create a figure for comparison
    fig, axes = plt.subplots(4, 3, figsize=(15, 16))
    fig.suptitle('Saliency Map Comparison: Standard vs. ReLU at Output', fontsize=16)

    # Target layer for visualization
    target_layer_std = [net_standard.conv2]
    target_layer_relu = [net_with_relu.conv2]

    # Initialize GradCAM for both models
    cam_std = GradCAM(model=net_standard, target_layers=target_layer_std)
    cam_relu = GradCAM(model=net_with_relu, target_layers=target_layer_relu)

    # Get predictions
    outputs_std = net_standard(images)
    outputs_relu = net_with_relu(images)
    _, predicted_std = torch.max(outputs_std, 1)
    _, predicted_relu = torch.max(outputs_relu, 1)

    for i in range(4):  # Process first 4 images
        # Original image
        img = images[i].cpu().numpy().transpose(1, 2, 0)
        img = img / 2 + 0.5  # unnormalize

        # Generate saliency maps for both models
        input_tensor = images[i].unsqueeze(0)

        # Standard model saliency
        grayscale_cam_std = cam_std(input_tensor=input_tensor, targets=None)
        grayscale_cam_std = grayscale_cam_std[0, :]
        vis_std = show_cam_on_image(img, grayscale_cam_std, use_rgb=True)

        # ReLU model saliency
        grayscale_cam_relu = cam_relu(input_tensor=input_tensor, targets=None)
        grayscale_cam_relu = grayscale_cam_relu[0, :]
        vis_relu = show_cam_on_image(img, grayscale_cam_relu, use_rgb=True)

        # Display original image
        axes[i, 0].imshow(img)
        axes[i, 0].set_title(f'Original: {classes[labels[i]]}')
        axes[i, 0].axis('off')

        # Display standard model saliency
        axes[i, 1].imshow(vis_std)
        axes[i, 1].set_title(f'Standard: {classes[predicted_std[i]]}')
        axes[i, 1].axis('off')

        # Display ReLU model saliency
        axes[i, 2].imshow(vis_relu)
        axes[i, 2].set_title(f'With ReLU: {classes[predicted_relu[i]]}')
        axes[i, 2].axis('off')

    plt.tight_layout()
    plt.subplots_adjust(top=0.95)
    plt.show()

# Generate side-by-side comparison
print("\nGenerating side-by-side comparison of saliency maps...")
compare_saliency_maps()
```

```
Generating side-by-side comparison of saliency maps...
```