Interpretable Classification of Real Waste Images Using Grad-CAM

1. About the Dataset

The Real Waste dataset consists of real-world images of waste materials classified into ten categories: Glass, Cardboard, Metal, Food Organics, Paper, Plastic, Miscellaneous Trash, Vegetation, and Textile Trash, as shown in the category visualization. The images are organized into folders, each named after a waste type, and loaded using torchvision.datasets.ImageFolder. Each image is color (RGB) and resized to 224×224 pixels as part of preprocessing.

Dataset Size: The dataset was split into 70% training, 10% validation, and 20% testing. Exact class counts are printed at dataset creation.

Column Types: The main data is image pixel data (three channels) and folder category labels.

2. About the Model

A custom CNN, SimpleCNN, was implemented with the following architecture:

- 5 convolutional blocks (increasing from 64 to 512 channels), each followed by ReLU activation and MaxPooling, capturing spatial hierarchies.
- Adaptive average pooling, flattening, and a single linear layer mapping to 10 waste classes.

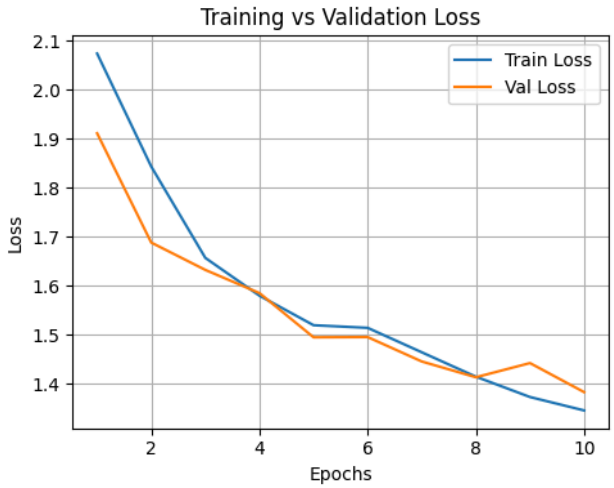
```
class SimpleCNN(nn.Module):
    def __init__(self, num_classes):
        super(SimpleCNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3,64,3,padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2,2),
            nn.Conv2d(64,128,3,padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2,2),
            nn.Conv2d(128,256,3,padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2,2),
            nn.Conv2d(256,512,3,padding=1), nn.ReLU(inplace=True),
            nn.Conv2d(512,512,3,padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2,2)
        )
        self.classifier = nn.Sequential(
            nn.AdaptiveAvgPool2d((1,1)),
            nn.Flatten(),
            nn.Linear(512, num_classes)
        )
    def forward(self,x):
        x = self.features(x)
        x = self.classifier(x)
        return x
```

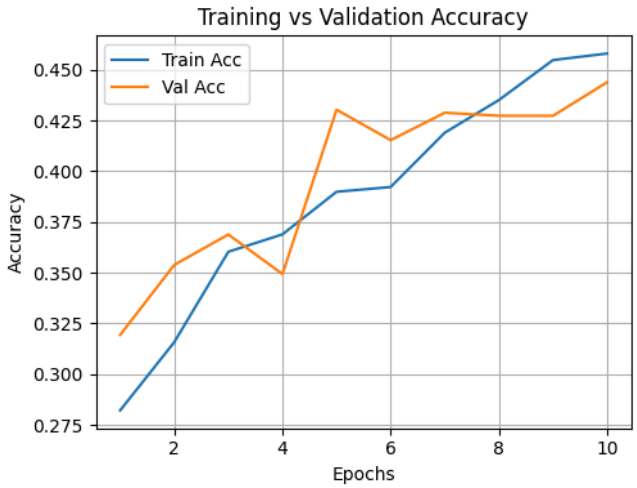
Training Details:

- **Optimizer:** Adam
- **Batch size:** 32
- **Epochs:** 10
- **Input size:** 224×224
- **Loss:** Steady improvement is observed; by epoch 10, validation loss is 1.38

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
1	2.0744	0.2822	1.9116	0.3193
5	1.5197	0.3899	1.4950	0.4303

10	1.3456	0.4579	1.3830	0.4438





3. About Grad-CAM

The Grad-CAM technique was implemented based on the reference PyTorch implementation, with code adaptation for this specific network. Grad-CAM enables visualization of the importance of input regions contributing to model predictions by utilizing gradients flowing into the last convolutional layer.

Implementation summary:

> Forward and backward hooks are attached to the target convolutional layer to capture activations and gradients.
>
> Given an input and a target class, gradients are computed via backward pass, and class activation maps are generated by weighting activation maps using the mean of their gradients.
>
> The result is upsampled on the original input for interpretation.

Key differences:

- It uses used the eighth conv layer (model.features) as the target layer.
- Implementation mostly follows reference, using a one-hot encoding for backpropagation and normalization of the output CAM.

Pseudocode:

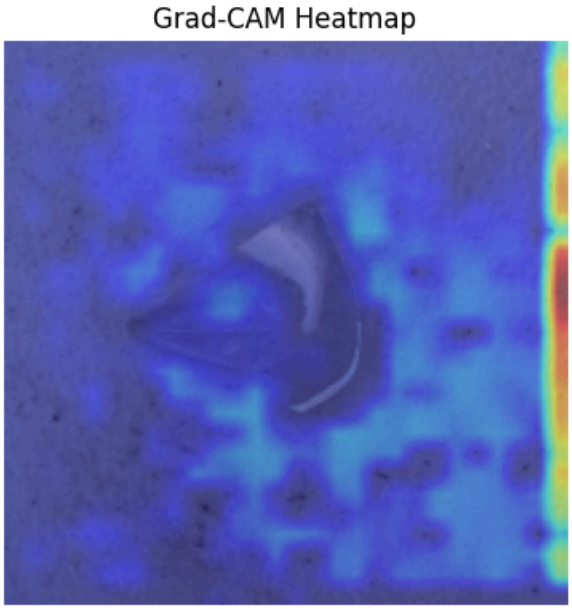
1. Register forward and backward hooks on the chosen conv layer.
2. Forward input through model, retain activations.
3. Compute gradients w.r.t. target class by backward.
4. Calculate channel-wise weights as mean of gradients.
5. Weighted sum of activations, apply ReLU, normalize.
6. Upsample and overlay on input image.

4. Model Explanation with Grad-CAM

Example of the model's interpretability using Grad-CAM, where both prediction and visualization are shown for a test image of 'Glass':

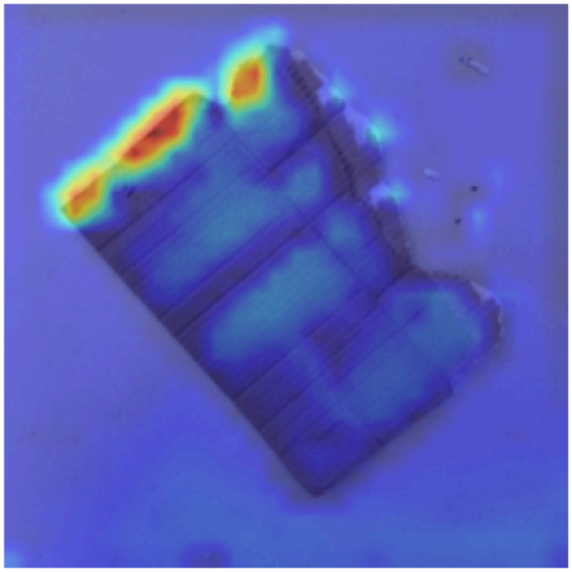
Example	True Label	Predicted	Grad-CAM Heatmap
1	Glass	Glass	IMAGE ATTACHED

The Grad-CAM heatmap aligns with the location of the glass object, validating that the model focuses on good features for classification rather than background noise. Our implementation iterates through test samples, and whenever the prediction matches the true class, overlays are generated to confirm interpretability.



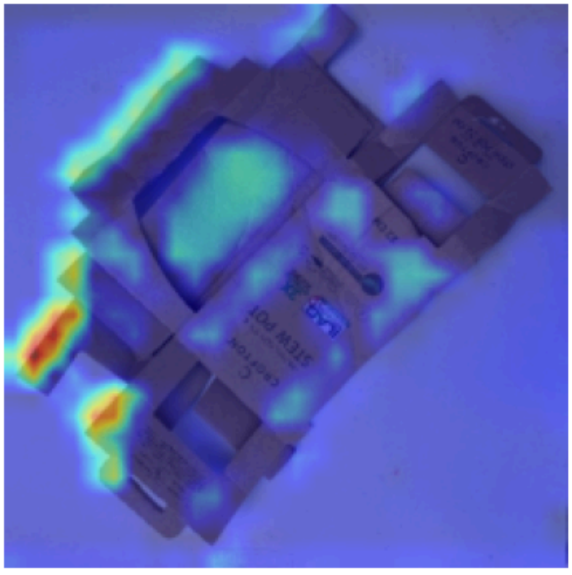
True: Cardboard
Pred: Cardboard

Grad-CAM Heatmap



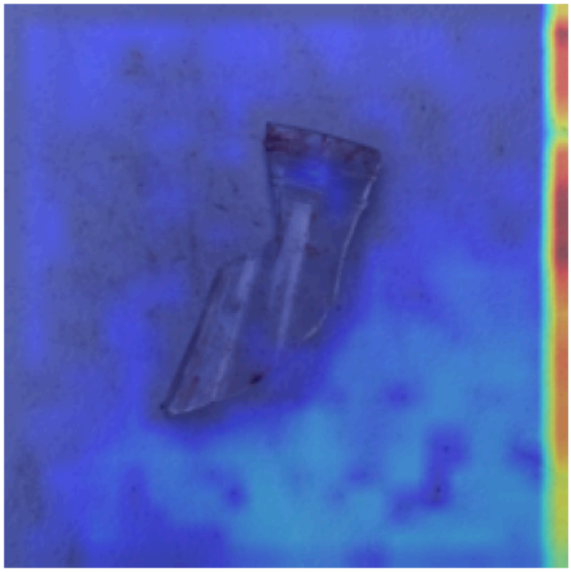
True: Cardboard
Pred: Cardboard

Grad-CAM Heatmap



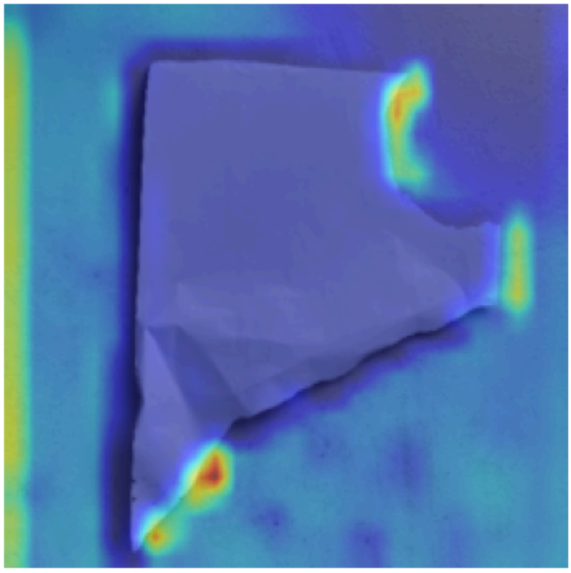
True: Glass
Pred: Glass

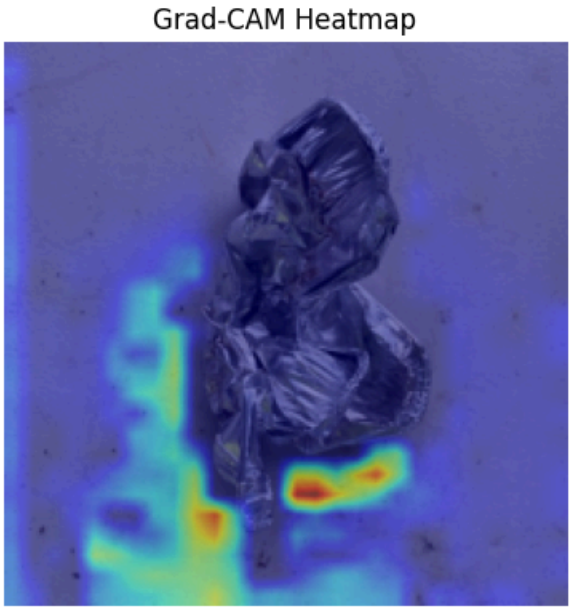
Grad-CAM Heatmap



True: Paper
Pred: Paper

Grad-CAM Heatmap





5. Key Takeaways from this assignment:

The Real Waste dataset provides a realistic image recognition task with a variety of waste categories.

Our CNN model was able to extract meaningful hierarchical features, delivering good accuracy even as a basic architecture. Using Grad-CAM helps us understand the model's decisions, showing that it focuses on the actual waste items rather than irrelevant parts of the image. It helps in interpreting.

The model could be improved by exploring deeper architectures or even pretrained ones, applying stronger data augmentation, addressing class imbalances, and experimenting with different layers for Gradcam visualization.