
PeliCAM

Manual



PeliCAM functions



Introduction

Explains what PeliCAM is and its key features.



Main Interface

Overview of the sidebar and the image viewer.



Core Functions

Load image, CAM, bounding box, LIME explanation, utilities.



Model & LIME Windows

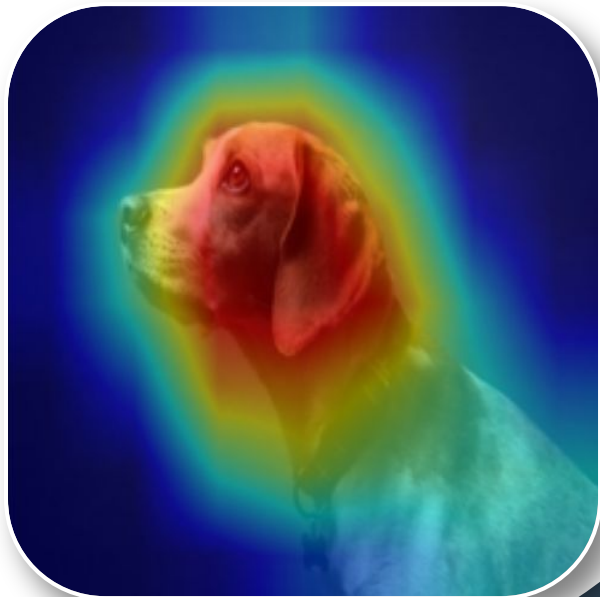
Custom model setup and LIME options and features.



Bounding Box Drawer

Drawing bounding boxes and comparison metrics.

What is PeliCAM?



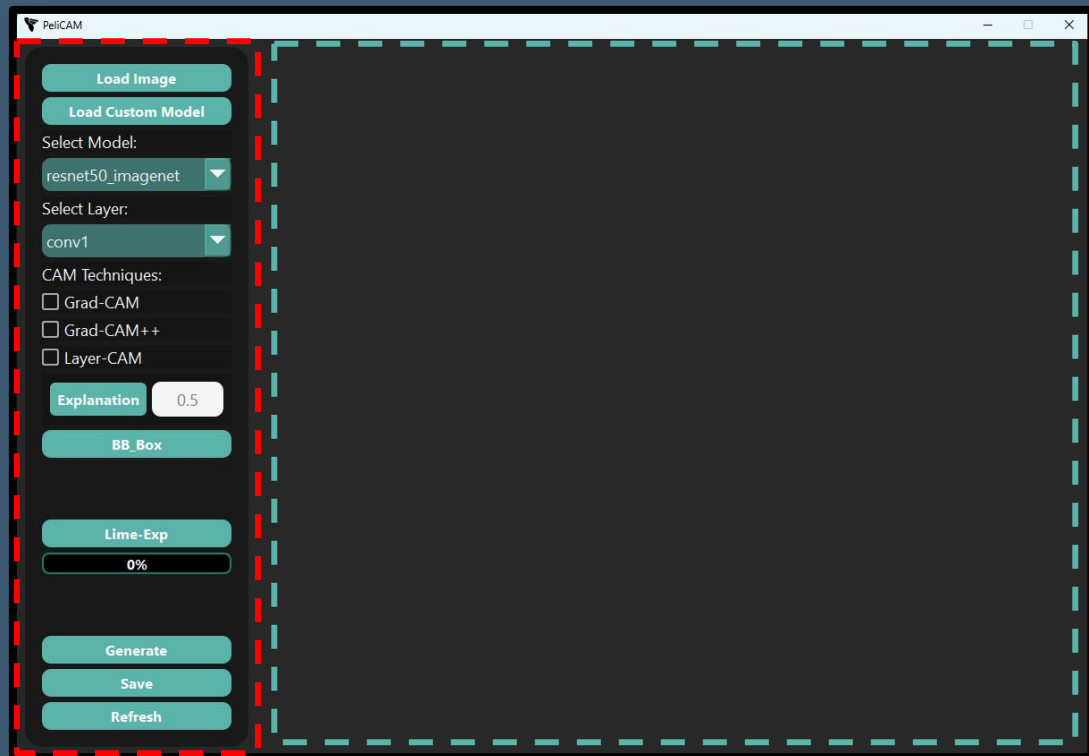
PeliCAM is a PyQt-based desktop app for visualizing deep learning model explanations using **CAM**, layer-based views, and **LIME**. It lets users load images, run predictions, and highlight key regions with heat maps and overlays.

A key feature of PeliCAM is its flexibility—you can integrate any custom PyTorch model, making it adaptable to various tasks and datasets. With interactive, tabbed outputs, it helps users better understand model behavior.





Main Window Overview



SIDEBAR:

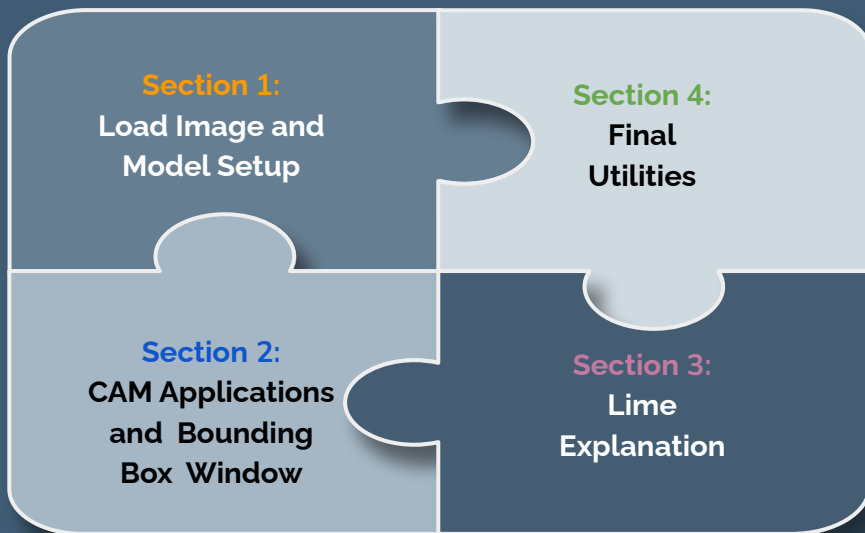
The sidebar contains all the interactive controls of the application, including buttons, checkboxes, and dropdown menus. These elements allow the user to configure settings, trigger actions, and manage the application's functionalities.

IMAGE VIEWER:

This area functions as a tabbed window where the output images and results are displayed. Each result opens in a separate tab, allowing easy navigation between multiple outputs.

SideBar Overview

The sidebar in PeliCAM serves as the main control panel of the application. It contains all the interactive elements needed to operate the app. Designed for ease of use, the sidebar allows users to configure settings, trigger processes, and customize outputs—all in one compact, accessible section of the main window.



The screenshot shows the PeliCAM sidebar interface, which includes the following elements:

- Load Image** button
- Load Custom Model** button
- Select Model:** dropdown menu showing **resnet50_imagenet**
- Select Layer:** dropdown menu showing **conv1**
- CAM Techniques:** checkboxes for **Grad-CAM**, **Grad-CAM++**, and **Layer-CAM**
- Explanation** button and a value input field showing **0.5**
- BB_Box** button
- Lime-Exp** button
- 0%** value input field
- Generate** button
- Save** button
- Refresh** button



Load Image and Model Setup

Load Image

Load Custom Model

Select Model:

resnet50_imagenet

Select Layer:

conv1

Select Model:

resnet50_imagenet

resnet50_imagenet

vgg16_imagenet

vgg19_imagenet

Select Layer:

conv1

layer4.0.conv1

layer4.0.conv2

layer4.0.conv3

layer4.0....nsample.0

layer4.1.conv1

layer4.1.conv2

layer4.1.conv3

layer4.2.conv1

layer4.2.conv2

layer4.2.conv3

This setup step prepares the image and model for generating visual explanation

This section includes two buttons and two dropdown menus for setting up the image and model.

Load Image: Import an image (.png, .jpeg, .jpg) for analysis.

Load Custom Model: Opens the Model Window to add a PyTorch model.

Select Model: Choose from default pretrained models (ResNet50, VGG16, VGG19) or any custom model added through the setup window.

Select Layer: Pick a convolutional layer from the selected model for applying CAM techniques.

Model Window



The Model Window in PeliCAM allows users to load and configure custom PyTorch models for use in the application. This is where you define the model architecture and load its trained weights.

Required Files:

To add a custom model, you need to provide:

Model Loader

Load .py File

No .py file loaded

Load .pth File

No .pth file loaded

Class Name:

Enter class name (e.g., CustomC...

OK

Python File (.py):

This file should define your model class

Model Weights File (.pt or .pth):

The trained weights file corresponding to the model architecture

The Class Name of the model is also need for getting the architecture.

example

```
import torch.nn as nn
import torch.nn.functional as F

class CustomCNN(nn.Module):
    def __init__(self, num_classes=2):
        super(CustomCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(32 * 56 * 56, num_classes) # Updated

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x))) # [B, 16, 112, 112]
        x = self.pool(F.relu(self.conv2(x))) # [B, 32, 56, 56]
        x = x.reshape(-1, 32 * 56 * 56) # Flatten
        x = self.fc1(x)
        return x
```



CAM & Bounding Box

CAM Techniques:

- ☒ Grad-CAM
- ☐ Grad-CAM++
- ☐ Layer-CAM

Explanation

0.8|

BB_Box

This section allows users to apply CAM-based explanations and use the bounding box utility which helps in refine explanations and mark regions of interest for further interpretation or annotation . It includes:

CAM Technique Checkboxes:

Select one or more CAM methods (e.g., Grad-CAM, Layer-CAM) to visualize model predictions on the selected layer.

Explanation Toggle with Threshold:

Applies a threshold to show only the most important activation regions. For example, a value of 0.8 highlights the top 20% of features, focusing on the most relevant parts of the image.

BB_Box Button:

Launches the Bounding Box Drawer Window, where users can manually or semi-automatically draw bounding boxes based on the CAM output.

Bounding Box Drawer

The Bounding Box Drawer allows users to manually mark ground truth regions on the image for comparison with model explanations. It includes the following features:

Color Selection (Dropdown)

Choose a box color that stands out from the image for clear visibility.

Drawing Boxes

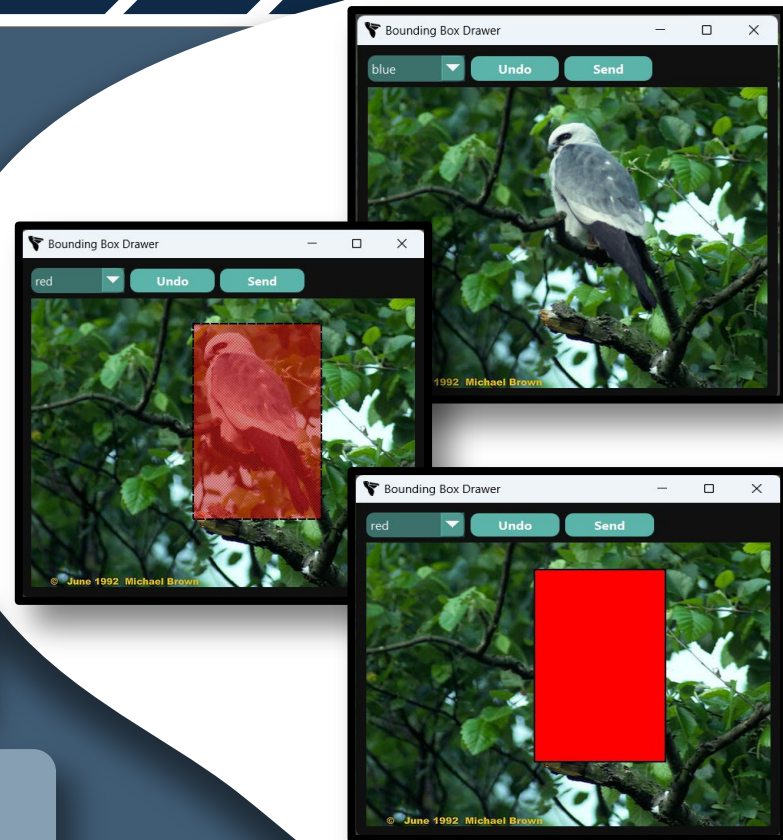
Draw ground truth bounding boxes by dragging the mouse with the left click. Multiple boxes can be added as needed.

Undo Button

Removes the most recently drawn bounding box. Useful for correcting mistakes without clearing all annotations.

Send Button

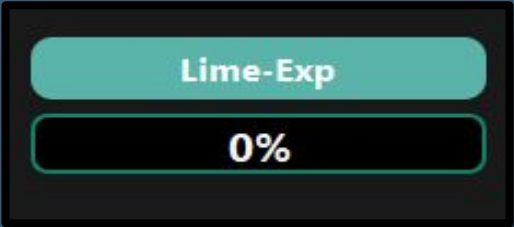
Works like the generate button—after boxes are drawn, it calculates quantitative comparison between ground truth and model explanations.





LIME Exp & Window

This section provides access to LIME-based model interpretability tools. It contains the LIME-Exp button, which opens a separate configuration window:



The image shows a dark-themed interface. At the top is a teal button with the text 'Lime-Exp' in white. Below it is a dark green progress bar with the text '0%' in white.

Lime-Exp

0%

LIME-Exp Button: Opens the LIME Window, where you can customize the type and detail of LIME explanations

Progress Bar: denotes the computation time for lime, hence it suggested to not do anything while the progress bar is loading.

Inside the LIME Window:



The image shows a window titled 'LIME-EXP' with standard window controls (minimize, maximize, close). Inside, there are three options with checkboxes: 'Heat-Map' (checked), 'Postive-only' (unchecked), and 'Both(+/-)' (checked). At the bottom, there is a label 'Num Features' followed by a slider set to the value '5'.

LIME-EXP

Explanation Options (Checkboxes)

Choose what kind of LIME output to display: **Heatmap** , **Positive Features** (regions supporting the prediction) , **Negative Features** (regions opposing the prediction). You can select one or multiple options.

Number of Features

Specify how many top features (superpixels) to show in the explanation. The default is 5, meaning the top 5 most influential regions (positive or negative) will be highlighted.

Final Utilities



Section 4 contains three essential buttons that manage the final steps of the explanation workflow. This section finalizes the explanation process and supports clean, repeatable workflows.

Generate

Save

Refresh

Generate Button

Generates all selected outputs—CAM visualizations, explanations, and LIME results—based on the current settings.

Save Button

Saves all images from the image viewer tabs into a user-selected folder. Each result (CAM, LIME, etc.) is saved as a separate image file.

Refresh Button

Clears all selections, loaded images, models, and outputs—effectively resetting the interface for a new session.

Note: LIME outputs will only be generated if the LIME Window is open and options are selected.





Image Viewer

The Image Viewer is the main display area of PeliCAM, designed to show all visual outputs in a tabbed format. Each image—whether it's the original input, a CAM result, or a LIME explanation—is displayed in its own tab for easy comparison and navigation.

Key features of the Image Viewer:

Tabbed Display

Each output (e.g., CAM heatmap, LIME overlay, saliency map) appears in a separate tab with a descriptive title. This allows users to switch between different visualizations effortlessly. Orange color indicates the current tab.

Integrated with Save and Copy Function

With right click on any image, we have option to save or copy the image

The Image Viewer provides a clear and organized way to interpret model outputs visually, supporting interactive exploration and side-by-side comparison

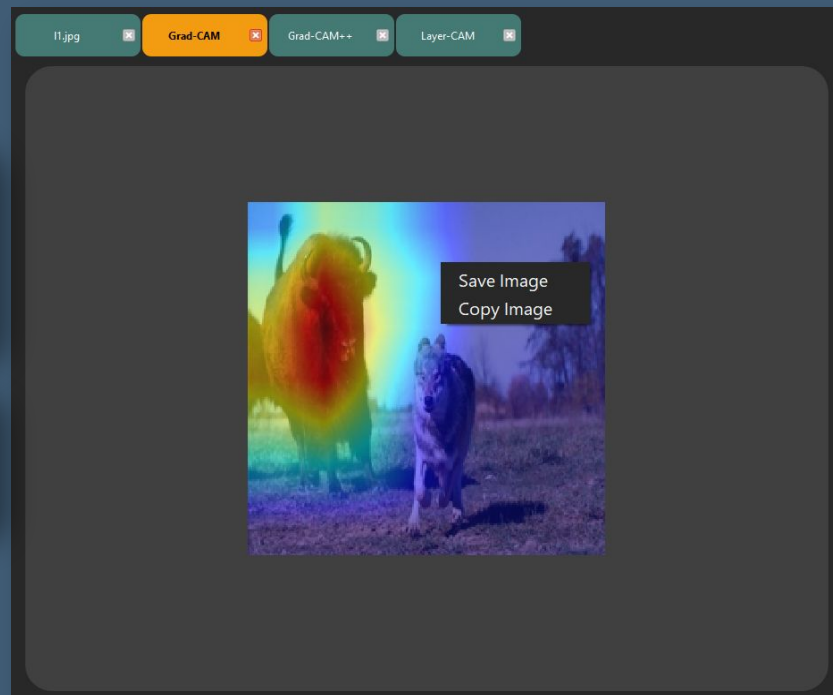
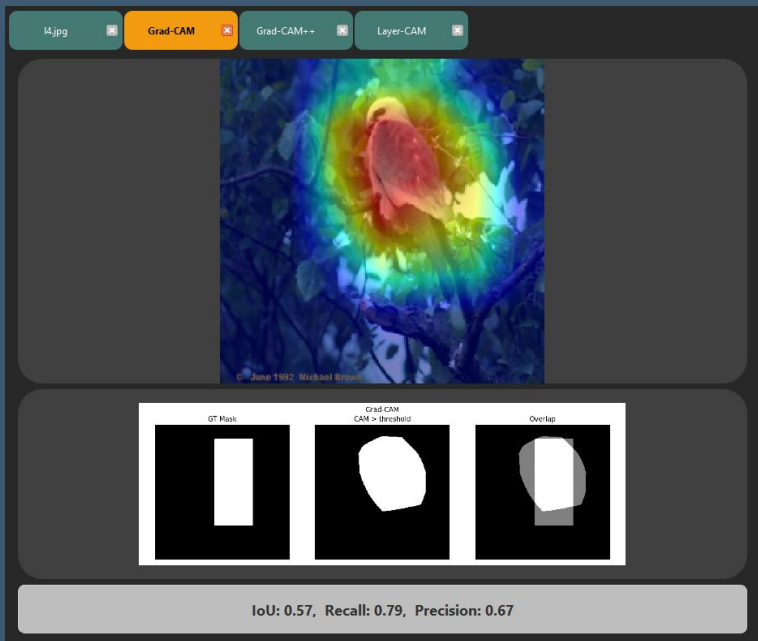


Image Viewer



This tab is automatically generated when the Send button is clicked from the Bounding Box Drawer Window. It provides both visual and quantitative analysis comparing ground truth regions to the CAM-based saliency map. These metrics help assess how well the model's highlighted regions align with the actual region of interest marked by the user.



Top Section – CAM Image

Displays the original CAM visualization used for comparison.

Middle Section – Region Plots

Ground Truth: Visual representation of the manually drawn bounding box.

Thresholded CAM Region: Highlights only the top features from the CAM based on the selected threshold (e.g., top 20%).

Overlap Visualization: Shows the intersecting area between the ground truth and CAM regions.

$$\text{IOU: } \frac{\text{CAM} \cap \text{Bbox}}{\text{CAM} \cup \text{Bbox}}$$

$$\text{Recall: } \frac{\text{CAM} \cap \text{Bbox}}{\text{Bbox}}$$

$$\text{Precision: } \frac{\text{CAM} \cap \text{Bbox}}{\text{CAM}}$$

FAQ

Q. What models does PeliCAM support?

PeliCAM supports both built-in models like ResNet50, VGG16, and VGG19, as well as custom PyTorch models. You can load your own `.py` file and `.pt/.pth` weights using the **Model Window**.

Q. Which image formats are supported?

You can load images in `.png`, `.jpeg`, or `.jpg` format for analysis and visualization.

Q. Why is my CAM or LIME output not showing?

- Selected the appropriate model and layer.
- Checked the required CAM or LIME options.
- Clicked the **Generate** button after making selections.
- For LIME, ensure the **LIME Window** is open and options are selected.

Q. What does the progress bar mean during LIME explanation?

The progress bar shows the computation status. Avoid interacting with the application while it's loading, as LIME may take a few seconds.

Q. Can I compare multiple outputs?

Yes! Each CAM, LIME, or input image appears in a separate tab within the **Image Viewer** for easy side-by-side comparison.





Thank You for using PeliCAM

We hope this manual helped you get started.