Deep Learning for Computer Vision
Lab 05

Bolutife Atoki
bolutife-oluwabunmi.atoki@etu.u-bordeaux.fr
Université de Bordeaux

Abstract

This paper contains my solution and results of the fifth lab session, resources used are listed at the end of the paper

1 Introduction

The aim of this session was to implement two Whitebox explainer models; GRADCAM[1] and FEM[2]. Whitebox explainer models delve into the internal workings of models to exploit the available knowledge of the network to create a better understanding of the prediction and the internal logic of the network. They help users understand how and why a model makes specific predictions, particularly in applications where model interpretability is crucial.

2 Methodology

2.1 Explainer models

Explainer models are models that provide insights in a form people are used to and understand, into the behavior and decision-making process of a trained model, with the aim of making the predictions or decisions of the trained model less complex, less interpretable models more transparent and understandable to humans.

2.2 Blackbox models

Black-box models make predictions based such that the internal workings of these models are not readily interpretable or understandable by humans. I.e the logic, rules, and processes used by black-box models are not transparent and are complex and inscrutable.

2.3 Whitebox explainer models

They help users understand how and why a model makes specific predictions, particularly in applications where model interpretability is crucial. The two models considered include:

2.3.1 GRADCAM

Gradient Class Activation Mapping (Grad-CAM) is a post-hoc explanation via visualization of class discriminative activations for a network. Grad-CAM leverages the structure of the CNN to produce a heat map of the pixels from the input image that con- tribute to the prediction of a particular class. Grad-CAM relies on the obervation that deeper convolutional layers of a CNN act as high-level feature extractors. So the feature maps of the last convolution layer of the network would contain the structural spatial information of objects in the image.
The features maps from the last convolution layer are not used directly by the method as they would contain information regarding all the classes present in the dataset. Thus, the method calculates the gradient of the output score for a particular class with respect to the features of the convolution layer.

2.3.2 FEM

Feature based Explanation Method (FEM) is similar to Grad-CAM as it also uses the observation that deeper convolutional layers of the network act as high- level feature extractors. FEM proposes that the final decision would be influ- enced by the strong features from k maps in the last conv layer. FEM supposes that the k maps have a Gaussian distribution and thus strong features from these maps would correspond to the rare features and uses K-sigma filtering to identify these strong and rare features.

3 Dataset

The dataset used is composed of images of two classes; **African Elephant** and **Black bear** in various environments with each having 51 images.

4 Implementation

The code implementation is made up of multiple script files having a main script file (**main.py**) to be called. The other files include:

1. **representations.py**: which holds the various representation functions used in the project.
   - **represent_heatmap()**:
     This takes in as parameters the saliency map and the desired colormap to be used , and returns the input saliency map represented as a heatmapped image according to the specified color map.
   - **represent_heatmap_overlaid()**:
     This takes in as parameters the saliency map, the image and the desired colormap to be used, and then applies the specified colourmap to the saliency map to get a heatmap (by calling the **represent_heatmap()**) function. The heatmapped saliency map is then blended with the input image by an alpha parameter that controls the blending, and finally returns this blended image.
2. **utils.py:**
   which holds the helper function used in the project.
   - **normalise()**:
     It takes in a matrix and returns its normalized version (to range 0 - 1) by dividing each element in it by its max value.
   - **grid_layout()**:
     It takes a list of images and a list of string titles as inputs, and plots & shows these images in a grid and saves the grid image.
   - **get_last_layer_name():**
     This takes in the model name and returns as a string the name of the last convolution layer for that pretrained model.
   - **get_required_size():**
     This takes in the model name and returns as the required image size for that pretrained model.
   - **get_decode_predictions_name():**
     This takes in the model name and returns the decode_predictions function for that model.
   - **get_preprocess_input_name():**
     This takes in the model name and returns the preprocess_input function for that model.
   - **get_model_():**
     This takes in the model name and returns the model and the last layer name as a string.
   - **make_classifier():**
     It takes in a string having the name of the blackbox model, creates and returns a pre-trained deep learning classifier model based on the specified model (Xception or ResNet).
3. **GradCAM.py**
   This script file holds the GRADCAM class and the class provides methods to compute and visualize activation maps highlighting the regions that are most important for a given class prediction. Its attributes are:
   - **model_name (str):** The name of the deep learning model used for prediction.
   - **img_array (numpy.ndarray):** The input image as a NumPy array.

   It has the following methods:
   - **get_decode_predictions():**
     It gets the appropriate decode_predictions function based on the model name.
   - **get_preprocess_input():**
     It gets the appropriate preprocess_input function based on the model name.

- **get_model():**
  It takes a model name, creates and returns the GradCAM model having the specified classifier, by calling a pretrained model based on the input, removes its softmax layer, gets the name of the last layer of that pretrained model, and then creates the Gradcam model which takes has as its input (the pretrained model's input), and as its outputs (the last layer's output and the pretrained model's output).
- **resize_array():**
  it takes as input the image array, resizes it according to the input size of the pretrained model, and returns it.
- **compute_gradients():**
  It takes in the Gradcam model and the class name of the object in the image and computes gradients of the class prediction with respect to the last convolutional layer, and returns it.
- **pool_gradients():**
  It takes the gradients (of dimension batch size x width x height x channels) as input and performing global average pooling for each channel reduces each to a 1x1 scalar, it then returns a list containing the 1x1 scalar for all channels.
- **weight_activation_map():**
  It takes the pooled gradients as input and then weights each activation layer by its corresponding pooled gradient value.
- **apply_relu():**
  It applies the ReLU activation function to the weighted activation maps by setting negative values to 0.
- **apply_dimension_average_pooling():**
  It applies average pooling along the channel dimension and returns the pooled array of size width x height.

4. **FEM.py**
   This script file contains the function for the FEM implementation. It has the following methods:

   - **get_img_array():**
     It loads an image from a file and converts it to a numpy array.
   - **expand_flat_values_to_activation_shape():**
     It expands a 1D array of values to the shape of a neural network activation map
   - **compute_binary_maps():**
     It computes binary maps based on feature maps
   - **aggregate_binary_maps():**
     It aggregates binary maps using the original feature map.
   - **compute_fem():**
     It computes Feature Extraction Maps (FEM) for an input image.

5. **main.py**
   It is the main python script for this project and it can be called directly from the command line, it has a main function which accepts the following arguments when called using argument parser;

   - path to test image
   - Index of the object in the image
   - The explanation method to be used
   - The name of the model to be used
   - the display type; **grid** which shows a grid of all results or **singles** which shows the results individually.

   An example format of calling the function is shown below

```
python main.py --test_image_path='ILSVRC2012_val_00018788.JPEG' ...
    --test_image_index=1 --explanation_method='GRADCAM' ...
    --model_name='Xception' --display_type='grid'
```

The main function does all the necessary imports of the utils, representations, GRADCAM and FEM files and then calls the class methods and functions for obtaining the saliency map (for the Explanation type and model type specified), representing it with a heatmap, overlaying the heatmap on the input image and finally displaying the obtained images either as a grid or individually, depending on the type specified with the grid display set as default.

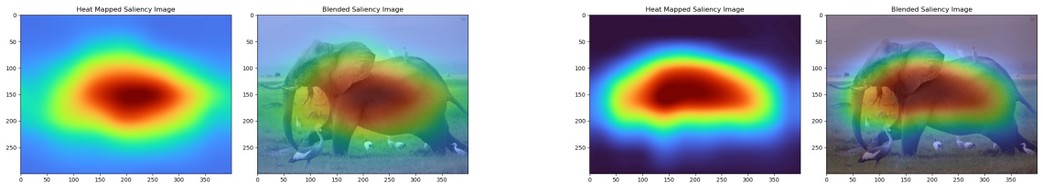
5 Results and Discussion

The results and subsequent discussions of the implementations are split into two; GRADCAM and FEM

5.1 GRADCAM Results

The GRADCAM explanation model was tested under different parameters; Explanation model type, Classifier model type, class index

5.1.1 Comparing results on same Image of African Elephant using ResNet and Xception models

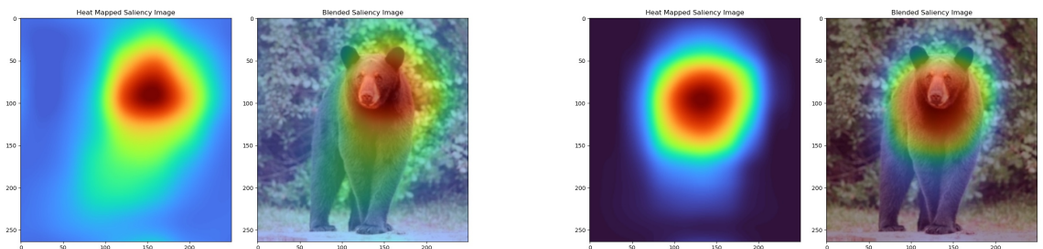


(a) GRADCAM explanation result using ResNet (b) GRADCAM explanation result using Xception

Figure 1: Comparison of explanation on African elephant image using GRADCAM with different models

Comparing both saliency maps, it is observed that the explanation using ResNet has more spread around the elephant in the image, while the explanation obtained using Xception has more depth, indicating a higher impact on the class than with the Resnet model. Overall however both explanation models performed well as they both included only the object and no background and these highlighted portions contain information from which features are extracted for classification.

5.1.2 Comparing results on same Image of Black bear using ResNet and Xception models



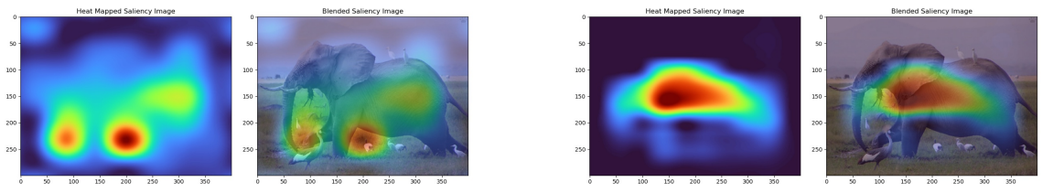
(a) Explanation result using ResNet (b) Explanation result using Xception

Figure 2: Comparison of explanation on Black bear image using different models

Comparing both saliency maps, it is observed that the explanation using ResNet has more spread around the elephant in the image, while the explanation obtained using Xception has more depth, indicating a higher impact on the class than with the Resnet model. Overall however both explanation models performed well as they both included only the object and no background and these highlighted portions contain information from which features are extracted for classification. The ResNet explanation is seen to highlight the limbs of the bear while the Xception model does not.

5.2 FEM Results

5.2.1 Comparing results on same Image of African Elephant using ResNet and Xception models

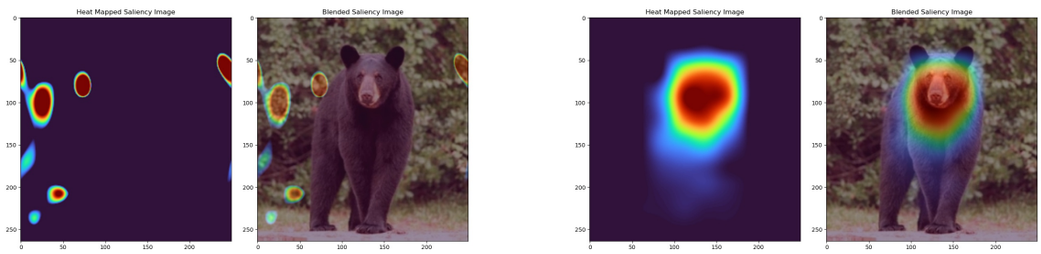


(a) Explanation result using ResNet (b) Explanation result using Xception

Figure 3: Comparison of explanation on African elephant image using different models

From the above figures, it is observed that the Explanation results using the two different classifier models highlight different parts of the object i.e both models use different features for their classification. It is seen that the ResNet model uses the feet, trunk and body (slightly) of the elephant for its classification, while the Xception model uses the ears, neck and fore limbs of the elephant for its classification. It is also observed from the saliency maps that the heatmap is generally hotter in the Xception explanation than in the ResNet model meaning that for the hotter places, the Xception model places more emphasis on these regions in the input image when making predictions. This may also suggest that the Xception model might be overfitting to specific patterns (regions) in the training data, which may not generalize well to unseen examples. and that the cooler ResNet model, may exhibit more robust generalization by considering a broader range of features.

5.2.2 Comparing results on same Image of Black bear using ResNet and Xception models

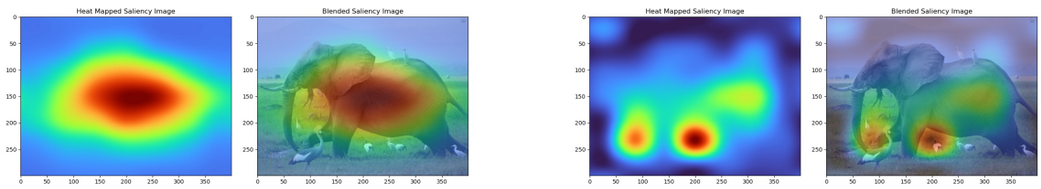


(a) Explanation result using ResNet (b) Explanation result using Xception

Figure 4: Comparison of explanation on Black bear image using different models

Here, it is observed that the ResNet explainer highlights portions in the background and nothing on the image, while the Xception explainer highlights the bear in the image. This could be because the differences in how these explainer models interpret and explain the predictions of the underlying classifier as FEM is class agnostic and GRADCAM is not, and FEM mostly highlights the rare portions in the image. Also while comparing the same bear image explanation for the ResNet model using FEM and GRADCAM, we see that GRADCAM explainer from Figure 2 highlights the Bear and not the background, so the ResNet model isn't the problem but the fact that FEM computes explanation without using the output class but with rare regions / features in the image.

5.2.3 Comparing results of same Image of African Elephant for GRADCAM and FEM Models (using ResNet for both)



(a) Explanation result using GRADCAM

(b) Explanation result using FEM

Figure 5: Comparison of explanation on African Elephant for GRADCAM and FEM Models

Here, it is observed that the highlighted features for both explanation methods differ even though the same classifier (ResNet) is used, these is because of the differences in how moth explanation methods compute their saliency map, so emphasis is on the gradients of the last convolution layer with respect to the output class for GRADCAM and emphasis is on rare features in FEM.

6 Conclusion

From the two explainer methods implemented (GRADCAM and FEM) and the various explanations gotten from them, it is observed that the explanation results rely on the methods of performing explanations by the explainer models and less on the classifier type, as opposed to what was observed during the last lab where the explanation results rely heavily on the explainer models, the stability, generalization of blackbox model used, the number of samples and lime feature parameters used for the LIME explainer, the size of the low res mask (for the RISE Explainer) etc.

References

[1] Fuad, K.A.A., Martin, P.E., Giot, R., Bourqui, R., Benois-Pineau, J. and Zemmari, A., 2020, November. Features Understanding in 3D CNNs for Actions Recognition in Video. In 2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA) (pp. 1-6). IEEE

[2] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. and Batra, D., 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626)