Deep Learning for Computer Vision
Lab 06

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

Abstract

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

1 Introduction

The aim of this session was to implement combine all the lab projects done and evaluate the results using techniques introduced in this lab. In more detail, the lab session involved integrating the transfer-learned ResNet model re-trained on the MexCulture dataset (from Lab 01) as the backbone classifier for all four explanation methods implemented in previous labs (GRADCAM[2], FEM[1], RISE, and LIME[3] from labs 4&5), and then evaluating the obtained explanations (saliency maps) on the two evaluation metrics (PCC and SIM) considered in Lab02 as well as the two metrics (Insertion and Deletion) introduced in this lab.

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.

2.4 Blackbox Explainer models

Blackbox explainer models are models that try to show the reasons behind a model's prediction or classification by highlighting the datapoints (in case of images; pixels) that are directly responsible for the model's output, without having access to the Model's Architecture or trained weights, such that the explainer model uses on the the input image as well as the classification score and the predicted class. The two models considered include:

2.4.1 LIME

LIME is a Feature attribution method which means that the method computes for each feature of an input sample its importance in the prediction. To do so, LIME uses perturbations of the considered sample and their corresponding (and perturbated) predictions to identify features of importance.

2.4.2 RISE

Randomized Input Sampling for Explanation (RISE) computes a saliency map emphasizing the features of interest for a given model prediction. It allows to identify the features that the model considers as important for the prediction of a given class (not necessarily the predicted one). The main idea of RISE is to generate a large number of random (binary) masks in low resolution and then to upscale them to the dimension of the input image (the upscaled mask values are therefore in [0, 1]). For each of these masks, a prediction is done by the model which produces a score for the considered class. While many methods consider the masked features as responsible for the perturbed prediction, the authors of RISE make the opposite assumption. In RISE, unmasked (or partially masked) features are considered as responsible for the prediction, and, instead of considering how far the perturbed prediction is from the initial prediction, RISE considers that important features are kept if and only if the perturbed prediction score is high.

3 Dataset

The dataset used is the Mexculture142 dataset which is made of images of Mexican Cultural heritage recorded during ANR PI Mexculture by IPN CITEDI and gaze fixations data recorded with an eye-tracker at LABRI UMR 5800 CNRS/University of Bordeaux/IPN. The dataset recorded gaze fixations of subjects executing a visual task of recognition of architectural styles of Mexican Cultural heritage. Each category represents different views of the same architectural structure.

The dataset contains 284 samples having 142 subclasses of Prehispanic, Colonial, Modern buildings, we provide 2 examples for each class and the corresponding .txt files of gaze fixations. Also, the saliency map of each image and .txt scanpath files where we have the coordinates and duration of fixations per subject.

The dataset contains 3 folders:
Images: contains 142 categories of Prehispanic, Colonial and Modern styles.
Fixations: holds .txt files which are the corresponding fixations of source images.
Density Maps: contains the subjective saliency maps of each category.

The identifier for each filename is composed as follows:
Images: SSS_XXX_YYY_N_#.png
Fixations: SSS_XXX_YYY_GazeFix_N_#.txt
Density Maps: SSS_XXX_YYY _GFDM_N_#.png

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. **constants.py**: which holds the various constant variables used in the project.
2. **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.

3. **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_image_index():**
     This takes in the image name and returns the class index for the image.

4. **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:** The transfer learned model to be explained and used for predictions.
   - **model_name (str):** The name of the deep learning model used for prediction.
   - **img_array (numpy.ndarray):** The input image as a NumPy array.
   - **class_index (str):** The name index of the object in the image to be classified.

   It has the following methods:

   - **get_model():**
     It takes a model name, creates and returns the GradCAM model having the specified classifier, using the passed in model, removes its softmax layer, gets the name of the last convolution layer of the 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).
   - **compute_gradients():**
     It takes in the Gradcam model and the class index of the object in the image and computes gradients of the class prediction with respect to the last convolutional layer, and returns it.
   - **compute_saliency_map():**
     It calls all the methods of the GRADCAM class in order, computes the saliency map 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.

5. **FEM.py**
   This script file holds the FEM class and provides methods to compute and visualize activation maps highlighting the regions that are most important for the models prediction. Its attributes are:

   - **model:** The transfer learned model to be explained and used for predictions.
   - **model_name (str):** The name of the deep learning model used for prediction.
   - **img_array (numpy.ndarray):** The input image as a NumPy array.
   - **class_index (int):** The name index of the object in the image to be classified.

   It has the following methods:

   - **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_saliency_map():**
     It computes FEM explanation for an input image and a classifier by removing the model's softmax layer, getting the name of the last convolution layer of the model, and creates the FEM 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), and uses the obtained feature / activation maps to obtain binary maps which are aggregated to become saliency maps.

6. **RISE.py**
   This script file holds the RISE class and provides methods to compute and visualize activation maps highlighting the regions that are most important for a given class' prediction

   - **model:** The transfer learned model to be explained and used for predictions.
   - **img_array (numpy.ndarray):** The input image as a NumPy array.
   - **class_index (int):** The name index of the object in the image to be classified.
   - **n_masks (int):** The number of masks to be generated to perturb the images.
   - **mask_size (int):** The size of the low res mask to be generated
   - **threshold(int):** The threshold value to set for having 1's in the generated binary mask.

   It has the following methods:

   - **generate_masks():**
     The function generates a set of random masks to perturb the image. These masks are created based on specified parameters, including the number of masks (n_masks), mask size (mask_size), and a threshold value (threshold). It is designed to create a set of random masks for perturbing an input image. This process involves resizing the original image to a square shape, ensuring consistency in dimensions. The user specifies the number of masks to generate (n_masks), the size of each mask (mask_size), and a threshold value (threshold) to determine the mask's pixel values. Random masks are generated with binary values, where the threshold decides whether a pixel in the mask should be set to 1 or 0. For each mask, the function randomly selects an origin point within the dimensions of an upsampled mask to ensure proper positioning during the perturbation process. These generated masks are then applied to the resized image, resulting in perturbed images where different regions of the image are selectively perturbed based on the mask patterns. The function returns a list containing both the perturbed images and the corresponding masks.
   - **obtain_prediction_scores():**
     It uses the perturbed_images (having dimensions batch_size, width, height, channels), and the class index for the object in the input image and makes predictions using the transfer learned model on the set of perturbed images. It then updates the object's field the prediction scores for the specified class.
   - **weight_saliency_maps():**
     It uses the prediction scores as weights and applies these weights to the generated upsampled masks, and then adds each weighted mask and finally divides it by the sum of predicted scores to obtain the saliency map.

- **calculate_saliency_map():**
  It uses the prediction scores as weights and applies these weights to the generated upsampled masks, and then adds each weighted mask and finally divides it by the sum of predicted scores to obtain the saliency map.
- **compute_saliency_map():**
  It calls all the methods of the RISE class in order, computes the saliency map and returns it.

7. **LIME.py**
   This script file provides methods to compute and visualize activation maps using LIME Explanation, highlighting the regions that are most important for a given class' prediction.
   It has the following methods:

   - **get_lime_explanation():**
     It instantiates and returns an explainer instance using the image, model, and lime parameters.
   - **explain_with_lime():**
     This is the main lime function called and it instantiates the explainer and generates both the mask and saliency map.

8. **evaluate.py**
   This script file provides methods to compute the different saliency map evaluation methods: Pearson's correlation coefficient (PCC), Similarity metric (SIM), Insertion algorithm and Deletion algorithm.
   It has the following methods:

   - **calculate_auc():**
     It calculates the area under the curve (AUC) using the trapezoidal rule.
   - **set_n_pixels_deletion():**
     It modifies the image array and its saliency map by setting the top n_pixels salient pixels to zero.
   - **predict_scores():**
     It predicts the scores for a given image using a model and returns the class probabilities.
   - **deletion():**
     It implements the Deletion algorithm to calculate deletion scores for an image array.
   - **set_n_pixels_insertion():**
     It modifies a blurred image array and its saliency map by setting the top n_pixels salient pixels to zero.
   - **deletion():**
     It implements the Insertion algorithm to calculate insertion scores for an image array.
   - **calculate_sim():**
     Calculates similarity between two saliency maps using minimum and sum scaling.
   - **calculate_pcc():**
     Calculates the Pearson Correlation Coefficient (PCC) between ground truth and a saliency map.

9. **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 the folder containing the test images
   - path to the folder containing the GDFM's
   - 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_images_folder_path='IPCV/UBx/DLCV/Lab06/Test_images' ...
    --test_gdfm_folder_path= ...
    'IPCV/UBx/DLCV/Lab03/MexCulture142/gazefixationsdensitymaps' ...
    --explanation_method='GRADCAM' --model_name='ResNet' ...
    --display_type='grid'
```

The main function does all the necessary imports of the utils, representations, evaluations, constants, GRADCAM, FEM, RISE, LIME 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, evaluating the obtained maps either by comparing with the groundtruth maps (with PCC and SIM) or by evaluating the standalone maps (with Insertion and Deletion) by calculating the mean and deviation of these metrics, and finally displaying the metrics & 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 evaluation of each of the Explanation methods are given below, along with some selected images having the obtained saliency maps, the groundtruth GDFM, the heatmapped saliency map, the saliency map blended with the input image, the insertion and deletion AUC's.

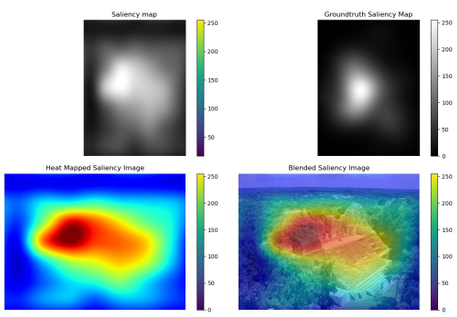
5.1 GRADCAM Evaluation

The GRADCAM explanation model was tested on 10 images using the transfer learned ResNet model for all three image classes and obtained results are presented below:

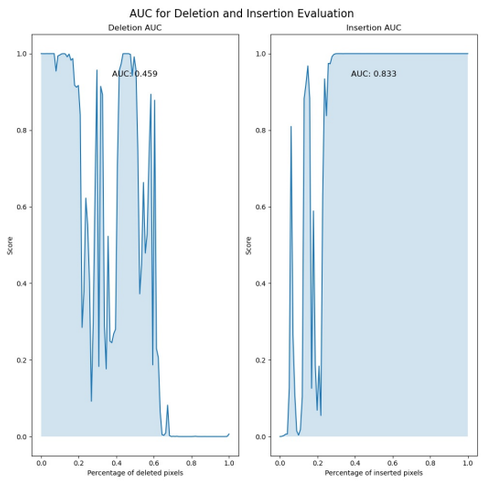
Table 1: Evaluation Metrics for GradCAM Saliency Maps

	PCC	SIM	Insertion	Deletion
GradCAM	0.5846 ± 0.186	0.5692 ± 0.09476	0.82376 ± 0.2857	0.4263 ± 0.2653

Sample Images are also provided below:

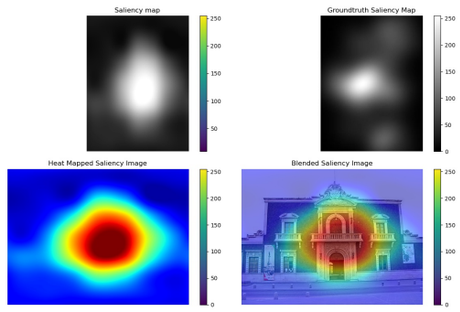


(a) GRADCAM explanation result using ResNet

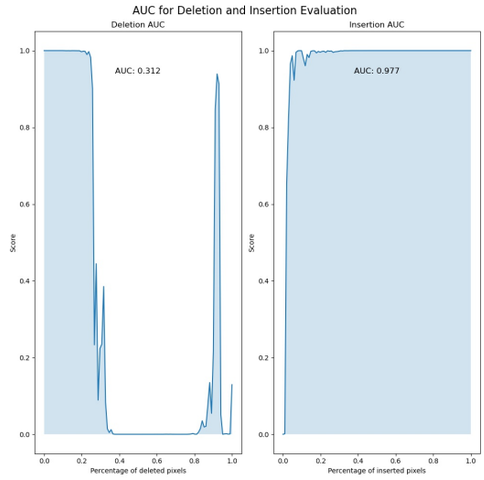


(b) GRADCAM explanation result evaluation

Figure 1: Visual and Analytical evaluation of GradCAM Explanation

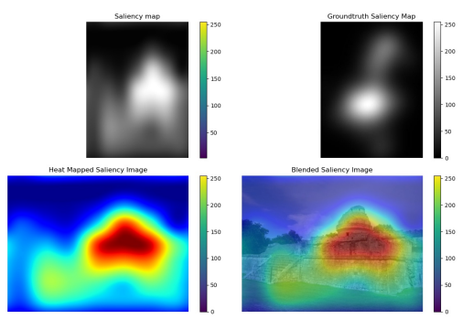


(a) GRADCAM explanation result using ResNet

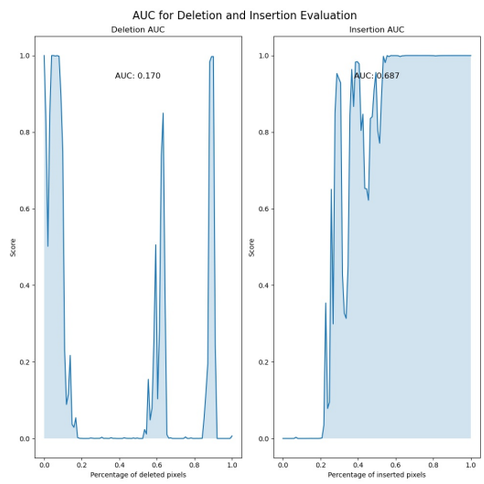


(b) GRADCAM explanation result evaluation

Figure 2: Visual and Analytical evaluation of GradCAM Explanation



(a) GRADCAM explanation result using ResNet



(b) GRADCAM explanation result evaluation

Figure 3: Visual and Analytical evaluation of GradCAM Explanation

From the table of results and observed saliency maps, it was noticed that:

- The mean and variance values of the Pearson Correlation Coefficient (PCC) metric shows moderately positive correlation between the saliency maps produced by the GradCAM model and the ground truth. Also that the consistency in the PCC values indicates that the model generally captures relevant features but may not perfectly align with the ground truth.
- The mean and variance values of the similarity metric indicates that the saliency maps have a reasonable level of similarity with the ground truth. However, the lower variance indicates a more consistent performance across different instances.
- The mean of the insertion AUC is high, implying that the model excels in correctly identifying relevant features, with a strong ability to discriminate between important and unimportant regions. However, the relatively high variance suggests that there are instances where the model struggles, leading to fluctuations in performance.
- The lower mean of the Deletion AUC when compared to Insertion AUC, indicates that the model may struggle with correctly ignoring irrelevant features, and the high variance indicates some inconsistencies in its ability to distinguish between important and unimportant regions, leading to a less robust performance in this aspect.
- Overall, after combining the PCC, SIM, Insertion AUC, and Deletion AUC metrics, a comprehensive view of the model's performance is obtained such that the PCC and SIM suggest a generally positive correlation and similarity, respectively, while the AUC metrics show the model's ability to highlight and suppress features appropriately.

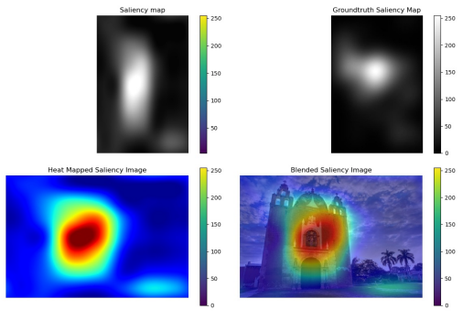
5.2 FEM Evaluation

The FEM explanation model was tested on 10 images using the transfer learned ResNet model for all three image classes and obtained results are presented below:

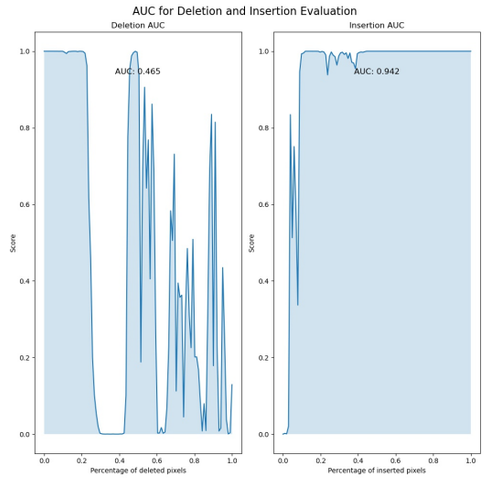
Table 2: Evaluation Metrics for FEM Saliency Maps

	PCC	SIM	Insertion	Deletion
FEM	0.5344 ± 0.1632	0.5901 ± 0.0862	0.7955 ± 0.297	0.5324 ± 0.2667

Sample Images are also provided below:

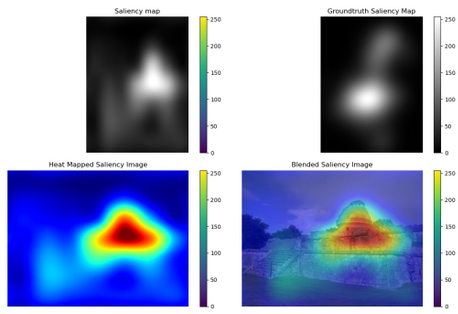


(a) FEM explanation result using ResNet

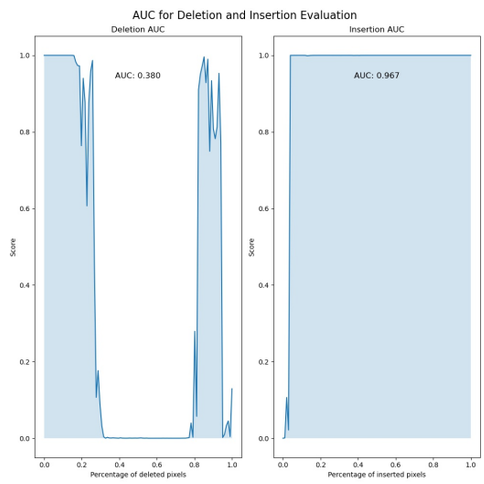


(b) FEM explanation result evaluation

Figure 4: Visual and Analytical evaluation of FEM Explanation

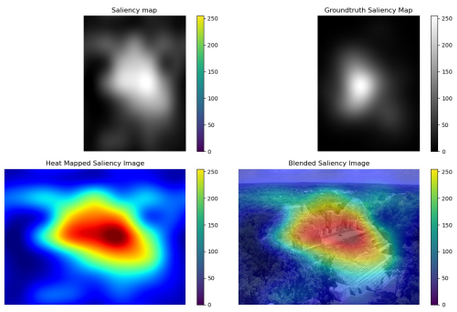


(a) FEM explanation result using ResNet

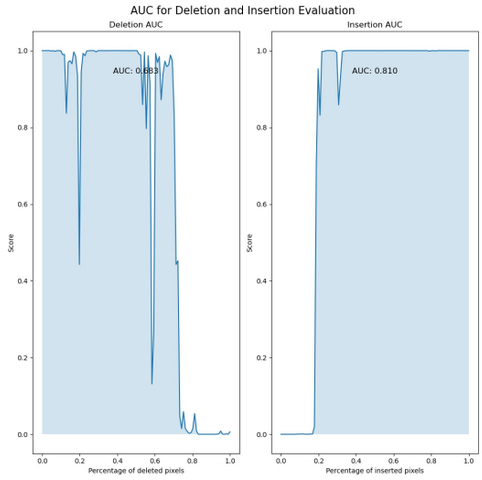


(b) FEM explanation result evaluation

Figure 5: Visual and Analytical evaluation of FEM Explanation



(a) FEM explanation result using ResNet



(b) FEM explanation result evaluation

Figure 6: Visual and Analytical evaluation of FEM Explanation

From the table of results and observed saliency maps, it was noticed that:

- The mean and variance values of the Pearson Correlation Coefficient (PCC) metric shows moderately positive correlation between the saliency maps produced by the GradCAM model and the ground truth. The model tends to capture relevant features, but the lower mean compared to the GradCAM model indicates a slightly weaker correlation.
- The mean and variance values of the similarity metric indicates that the saliency maps have a reasonable level of similarity with the ground truth. The higher mean compared to GradCAM suggests a stronger overall similarity.
- The mean of the insertion AUC is high, implying a strong ability to correctly identify relevant features, similar to GradCAM. However, the relatively high variance indicates instances where the model's performance is less consistent in discriminating between important and unimportant regions.
- The lower mean of the Deletion AUC when compared to Insertion AUC, indicates that the model could have challenges in correctly suppressing irrelevant features, and the high variance indicates some inconsistencies in its ability to distinguish between important and unimportant regions, leading to a less robust performance in this aspect.
- Overall, after combining the PCC, SIM, Insertion AUC, and Deletion AUC metrics, a comparable performance to GradCAM is observed, with strengths in capturing relevant features (as indicated by higher Insertion AUC) but potential weaknesses in suppressing irrelevant features (lower Deletion AUC). The SIM values suggest a generally higher similarity to the ground truth compared to GradCAM.

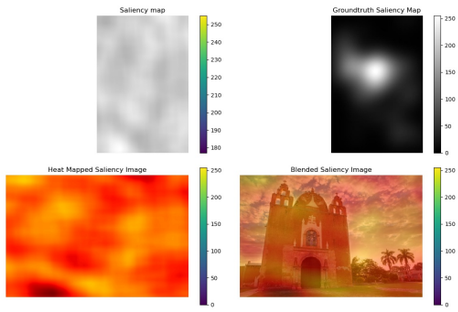
5.3 RISE Evaluation

The RISE explanation model was tested on 10 images using the transfer learned ResNet model for all three image classes and obtained results are presented below:

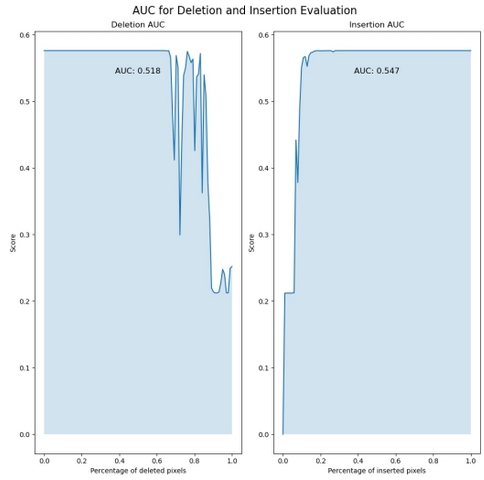
Table 3: Evaluation Metrics for RISE Model Saliency Maps

	PCC	SIM	Insertion	Deletion
RISE	-0.0033 ± 0.1587	0.3363 ± 0.05295	0.4676 ± 0.1054	0.4100 ± 0.1306

Sample Images are also provided below:

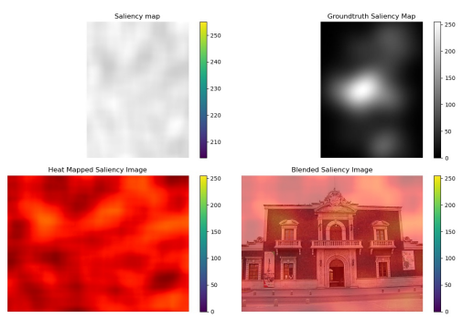


(a) RISE explanation result using ResNet

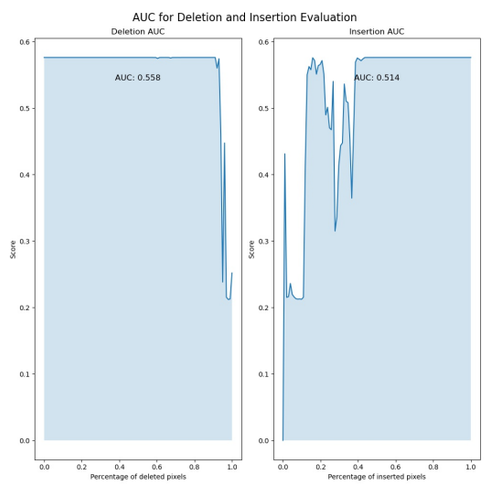


(b) RISE explanation result evaluation

Figure 7: Visual and Analytical evaluation of RISE Explanation

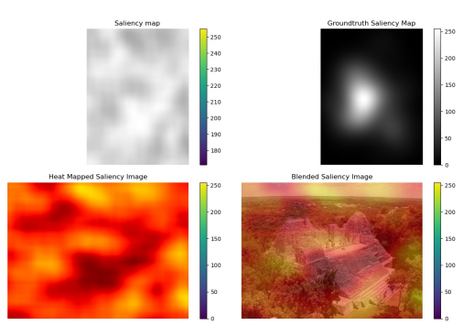


(a) RISE explanation result using ResNet

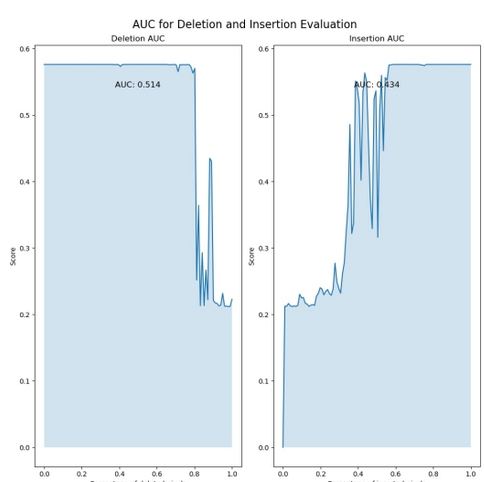


(b) RISE explanation result evaluation

Figure 8: Visual and Analytical evaluation of RISE Explanation



(a) RISE explanation result using ResNet



(b) RISE explanation result evaluation

Figure 9: Visual and Analytical evaluation of RISE Explanation

From the table of results and observed saliency maps, it was noticed that:

- The negative mean of the Pearson Correlation Coefficient (PCC) metric suggests a lack of a clear linear correlation between the saliency maps and the ground truth, while the variance indicates a degree of inconsistency, emphasizing challenges in aligning with the ground truth.
- The mean and variance values of the similarity metric indicates a lower level of similarity between the saliency maps and the ground truth when compared to GradCAM and FEM

models. The lower mean suggests a weaker overall resemblance, and the low variance implies relatively consistent performance across different instances.

- The mean of the Insertion AUC suggests a moderate ability to correctly identify relevant features, but the variance indicates some variability in performance. The lower mean when compared to previous models point to some challenges in capturing important features consistently.
- The mean of the Deletion AUC indicates a moderate ability to suppress irrelevant features, but again, the variance indicates variability in performance. The model is prone to facing challenges in consistently distinguishing between important and unimportant regions in certain instances.
- Overall, after combining the PCC, SIM, Insertion AUC, and Deletion AUC metrics, the RISE model demonstrates unique characteristics compared to previous models. The negative mean in PCC and relatively low SIM values suggests less linear correlation and lower overall similarity to the ground truth. While the AUC metrics indicate a moderate but inconsistent ability to identify and suppress features.

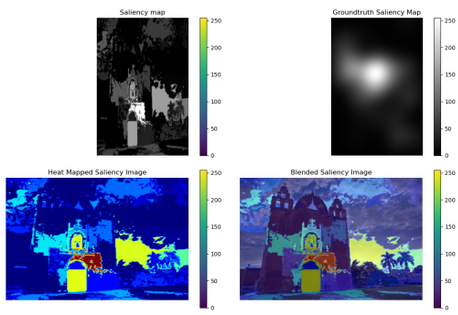
5.4 LIME Evaluation

The LIME explanation model was tested on 10 images using the transfer learned ResNet model for all three image classes and obtained results are presented below:

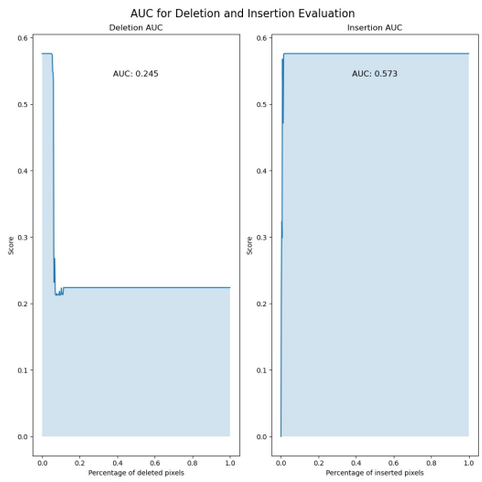
Table 4: Evaluation Metrics for LIME Saliency Maps

	PCC	SIM	Insertion	Deletion
LIME	0.0750 ± 0.2154	0.0145 ± 0.0552	0.4958 ± 0.1418	0.3751 ± 0.1533

Sample Images are also provided below:

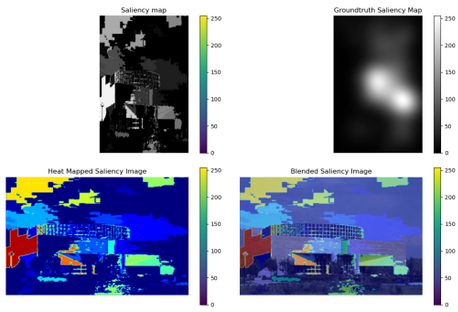


(a) LIME explanation result using ResNet

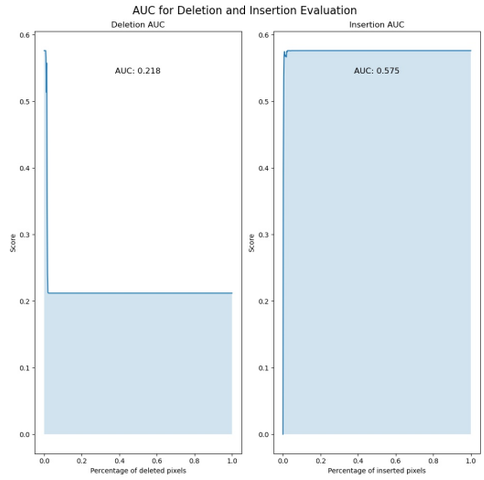


(b) LIME explanation result evaluation

Figure 10: Visual and Analytical evaluation of LIME Explanation

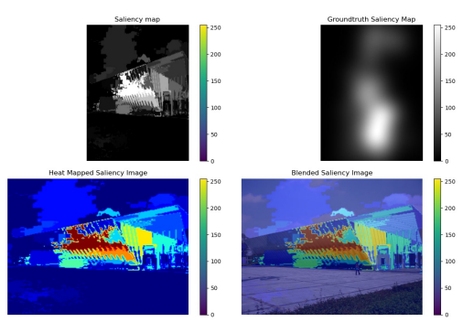


(a) LIME explanation result using ResNet

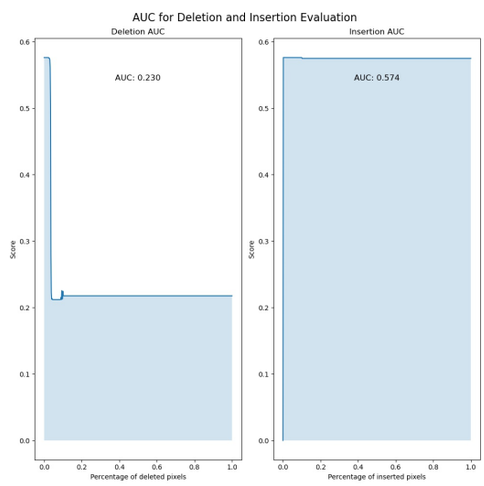


(b) LIME explanation result evaluation

Figure 11: Visual and Analytical evaluation of LIME Explanation



(a) LIME explanation result using ResNet



(b) LIME explanation result evaluation

Figure 12: Visual and Analytical evaluation of LIME Explanation

From the table of results and observed saliency maps, it was noticed that:

- The positive mean of the Pearson Correlation Coefficient (PCC) metric suggests a weak positive correlation between the saliency maps and the ground truth. However, the high variance indicates substantial inconsistency in aligning with the ground truth, indicating challenges in capturing relevant features consistently.
- The mean and variance values of the similarity metric indicates a very low level of similarity between the saliency maps and the ground truth, as both the low mean and variance suggest that the model faces difficulties in generating saliency maps that closely resemble the ground truth.
- The mean of the Insertion AUC suggests a moderate ability to correctly identify relevant features, but the variance indicates some variability in performance, highlighting that the model's ability to discriminate between important and unimportant regions is inconsistent.
- The mean of the Deletion AUC indicates a moderate ability to suppress irrelevant features, but again, the variance indicates variability in performance. The model is prone to facing challenges in consistently distinguishing between important and unimportant regions in certain instances.
- Overall, after combining the PCC, SIM, Insertion AUC, and Deletion AUC metrics, the AUC metrics indicate a moderate but inconsistent ability to identify and suppress features, while the high variance in PCC, SIM, and AUC metrics suggests a need for targeted improvements in the model.

6 Conclusion

The objectives of the lab session which were to learn to integrate separate blocks of codes, implement new saliency map evaluation techniques, replace the Pretrained model being used for the previous labs with the model transfer learned from the first lab, and to evaluate the four explanation methods taught in all the labs on the MexCulture dataset, were all carried out, with the results well documented and discussed above. The obtained results from which we can conclude by saying that the GradCAM explainer model tested the best in all metrics but the SIM (which it came second in), while the LIME model had the least results.

It should be noted that this project took several hours of problem formulation, coding and report writing, with extra days spent web-surfing and debugging an error obtained while trying to calculate the gradients for the GradCAM model. The error was due to the fact that the gradient_tape function of tensorflow can only watch tensor for which it has at least one of the inputs to the model, and with the way the model was setup during the transfer learning process in the first lab, the ResNet Backbone model was added as a layer to the model, hence gradient_tape couldn't access its layers and was returning an error, until this was eventually noticed and the model architecture was changes, model re-trained and this time the layers of the backbone were imported as layers into the new model.

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)

[3] Marco, T.R, Sameer S., Carlos G., 2016. Should I Trust You?": Explaining the Predictions of Any Classifier