Explainable AI - Project 1

Sebastiano D'Arconso (VR489066) - Tommaso Del Prete (VR488382)

February 15, 2024

Abstract

This project aims to implement a 2D Convolutional Neural Network (CNN) for image classification and then to use two eXplainable AI algorithms to extract the most important areas from the images. After that, a statistical analysis is performed on the final attributions. The eXplainable AI methods used in this project are the Integrated Gradients method (one implemented by us and one provided by the Captum library) and the LIME attribution method for post-hoc analysis. The dataset used is the Muffin vs Chihuahua dataset. The statistical analysis relies on the study of the histogram plots of each attribution and the correlations between each attribution.

1 Introduction

In this project we had to:

- Select a dataset for image classification
- Implement a Convolutional Neural Network
- Implement the Integrated Gradients algorithm
- Perform a post-hoc analysis with the Integrated Gradients method provided by the Captum library
- Perform post-hoc analysis using LIME (or SHAP) attribution method
- Compare the attributions extracted with the methods
- Comment the results

2 Dataset - Muffin vs Chihuahua

This dataset is composed of around 6000 images (5917) of chihuahuas and muffins. The images are scraped from google images and the count is around 2600 for muffins and 3400 for chihuahuas, one first basic preprocessing is performed in order to convert the non-RGB images in RGB and to assign the correct label to each photo (0 for muffins and 1 for chihuahuas). We then calculated the mean and standard deviation for the RGB values of each image in order to use those values later for the transformations.

2.1 Stratified K-Fold

To implement the k-fold subdivision, we utilized the stratified k-fold function from the scikit-learn library, dividing the dataset into five distinct folds. The decision to employ five folds was influenced by considerations of available resources, as well as the common practice of using 5 or 10 folds for "small" datasets. This cross-validation technique is a modified version of k-fold that yields stratified folds, preserving the proportion of samples for each class within the folds. Subsequently, we trained the model on each fold, comparing the results to identify the model with the best validation accuracy. This process allowed us to calculate the average performance of the model across all folds.

3 Model - CNN

We opted for a 3-layer deep Convolutional Neural Network (CNN) for our model, incorporating batch normalization and max pooling in each layer, excluding the final classification layer. The training process spanned 50 epochs on each of the 5 folds. Throughout each fold, we monitored and recorded losses and accuracy at each epoch. The model was saved at each epoch where it demonstrated superior validation accuracy, resulting in the creation of five distinct models, each corresponding to a fold in the dataset.

4 Post-hoc analysis - Integrated Gradients

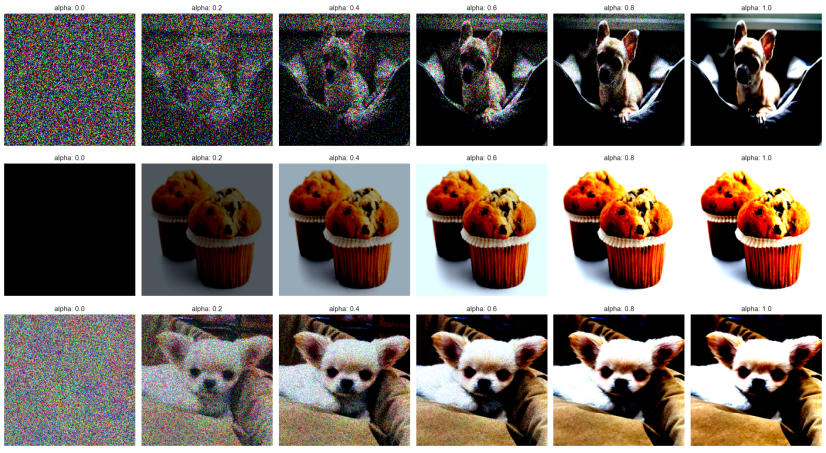
Integrated gradients represent the integral of the gradients with respect to inputs along the path from a given baseline to output. The integral can be approximated using a Riemann Sum. In the implementation we used the Trapezoidal rule.

$$IntegratedGrads_i^{approx}(x) ::= \overbrace{(x_i - x_i')}^{5.} \times \overbrace{\sum_{k=1}^{m}}^{4.} \frac{\partial \overbrace{F(x' + \overbrace{\frac{k}{m}}^{1.} \times (x - x'))}^{2.}}{\partial x_i}}^{3.} \times \overbrace{\frac{1}{m}}^{4.}$$

1. Generate alphas α
2. Generate interpolated images $= (x' + \frac{k}{m} \times (x - x'))$
3. Compute gradients between model F output predictions with respect to input features $= \frac{\partial F(\text{interpolated path inputs})}{\partial x_i}$
4. Integral approximation through averaging gradients $= \sum_{k=1}^{m}$ gradients $\times \frac{1}{m}$
5. Scale integrated gradients with respect to original image $= \sum_{k=1}^{m}$ gradients $\times \frac{1}{m}$. The reason this step is necessary is to make sure that the attribution values accumulated across multiple interpolated images are the same units and faithfully represent the pixel importance on the original image.

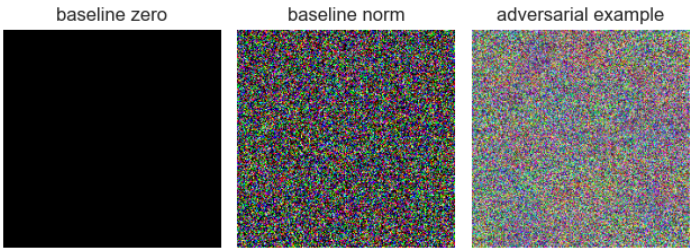
Each of the parts of the integrated gradients formula have been implemented separately and then merged together to perform the algorithm and can be seen in the **utils.py** file.

Example of interpolated images, first with normal baseline, then with zero baseline and the last with adversarial baseline (baseline with 50/50 output from the model):



4.1 Baseline choices

To objectively identify the pixels crucial for predicting our label, we employ a baseline image for comparison. An effective baseline should contain neutral or uninformative pixel features, and various baselines can be employed. In our experiments, we tested a black baseline, a noise baseline, and a baseline determined using a method designed to find a completely uninformative image. We termed the latter the "adversarial baseline" because it is derived through an adversarial approach. In this method, we set a goal and adjust the values of a random image for a fixed number of steps until the model scores 50/50 on that image. In the instance presented in this report, we initiated the process with a noise baseline, but the code allows us to showcase how results vary when using a random image from the dataset.

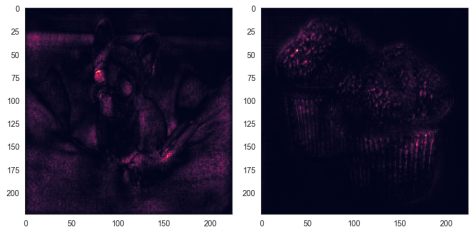


5 Post-hoc analysis - LIME

LIME, an acronym for Local Interpretable Model-Agnostic Explanations, is a post hoc, perturbation-based explainability technique. In elucidating the rationale behind a model's specific prediction for a given input instance, the LIME algorithm operates by presenting numerous slightly perturbed examples of the original input to the model. It then observes how the model's predictions change in response to these minor input modifications. These perturbations are introduced at the feature level of the input instance, involving modifications to pixels and pixel regions to generate new perturbed inputs. Consequently, the pixels or pixel regions that exert the most significant influence on the model's prediction are highlighted, offering insights into their impact on the model's predicted output for the given input instance.

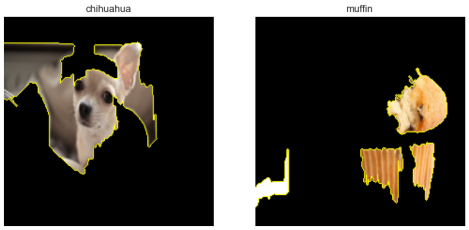
6 Attributions

In **gradient based methods** such as integrated gradients the goal is to estimate the attribution maps. An attribution map captures the importance of each input feature (in our case, pixels) for a specific output class.



(a) Chihuahua (b) Muffin

In **perturbation methods**, such as LIME, the method provides explanations for individual predictions by highlighting the important regions (superpixels: small, non-overlapping regions) in the input image that influenced the model's decision. The output of LIME for an image classification model like our case is typically a set of superpixels with assigned weights, indicating their influence on the model's prediction.

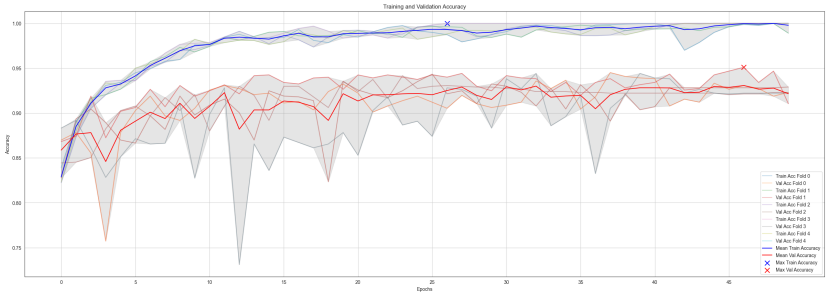


7 Results

After the training we performed k-fold cross validation to obtain the mean accuracy of the model as well as the F1 score across the k-folds.

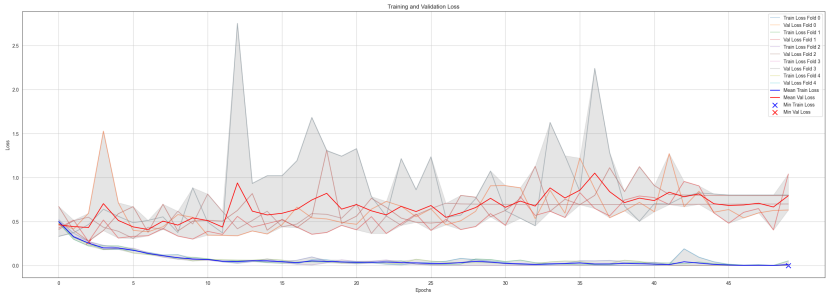
7.1 Accuracy

Max train	Mean train	Max val	Mean val
100%	97%	95%	91%



7.2 Loss

Min train	Mean train	Min val	Mean val
0.0	0.064	0.272	0.664

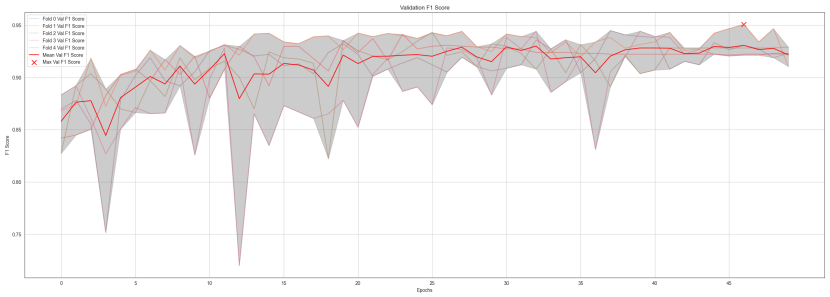


7.3 F1 score

For calculating the F1 score we used the f1 score function provided by sklearn.

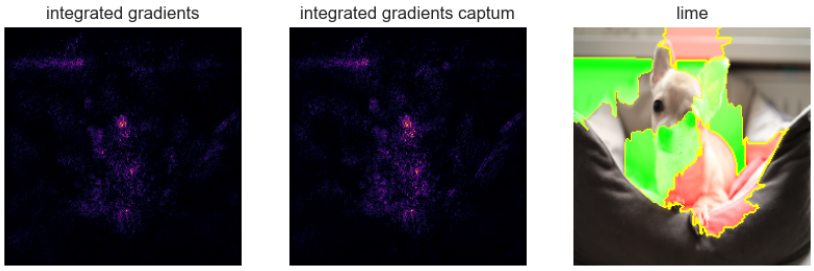
Max/mean	0	1	2	3	4
Max	0.945	0.951	0.943	0.941	0.944
Mean	0.911	0.928	0.911	0.915	0.892

Total mean F1 score across the folds: 0.912.



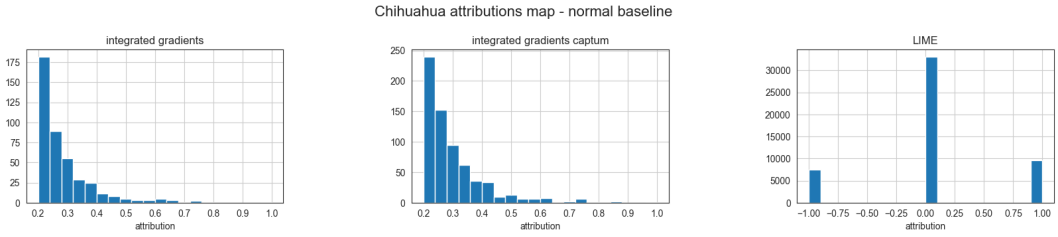
7.4 Statistical analysis

Subsequently, we conducted a statistical analysis on the attribution maps to evaluate distinctions and similarities among the three methods. The analyses presented in this context relate to the attribution maps generated through the application of Integrated Gradients using our approach and captum, employing the "normal" (random) baseline. The accompanying plots serve as visual references.



7.4.1 Analysis 1: histogram

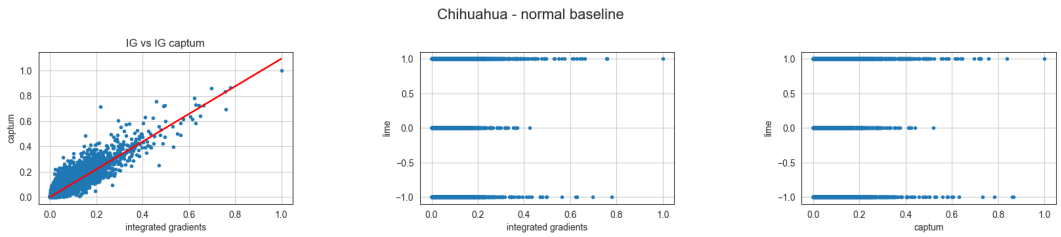
For this analysis, we flattened each attribution map (normalized between 0 and 1) and represented it as a histogram. This approach allows us to visualize the concentration and gradient values for each pixel. Notably, through empirical observation, we identified gradients with values exceeding 0.2 as particularly informative for our analysis. Consequently, we chose to construct the histogram exclusively for gradients with values greater than or equal to 0.2. This selection yields more informative plots, as plotting the entire flattened attribution map would result in a majority of pixels with values close to zero, making it challenging to compare the informativeness of each method. In the case of LIME, a similar approach was employed. Given that the attribution map is a mask with values of 0, 1, or -1, we flattened the mask and analyzed the concentration of pixels for values 1 and -1.



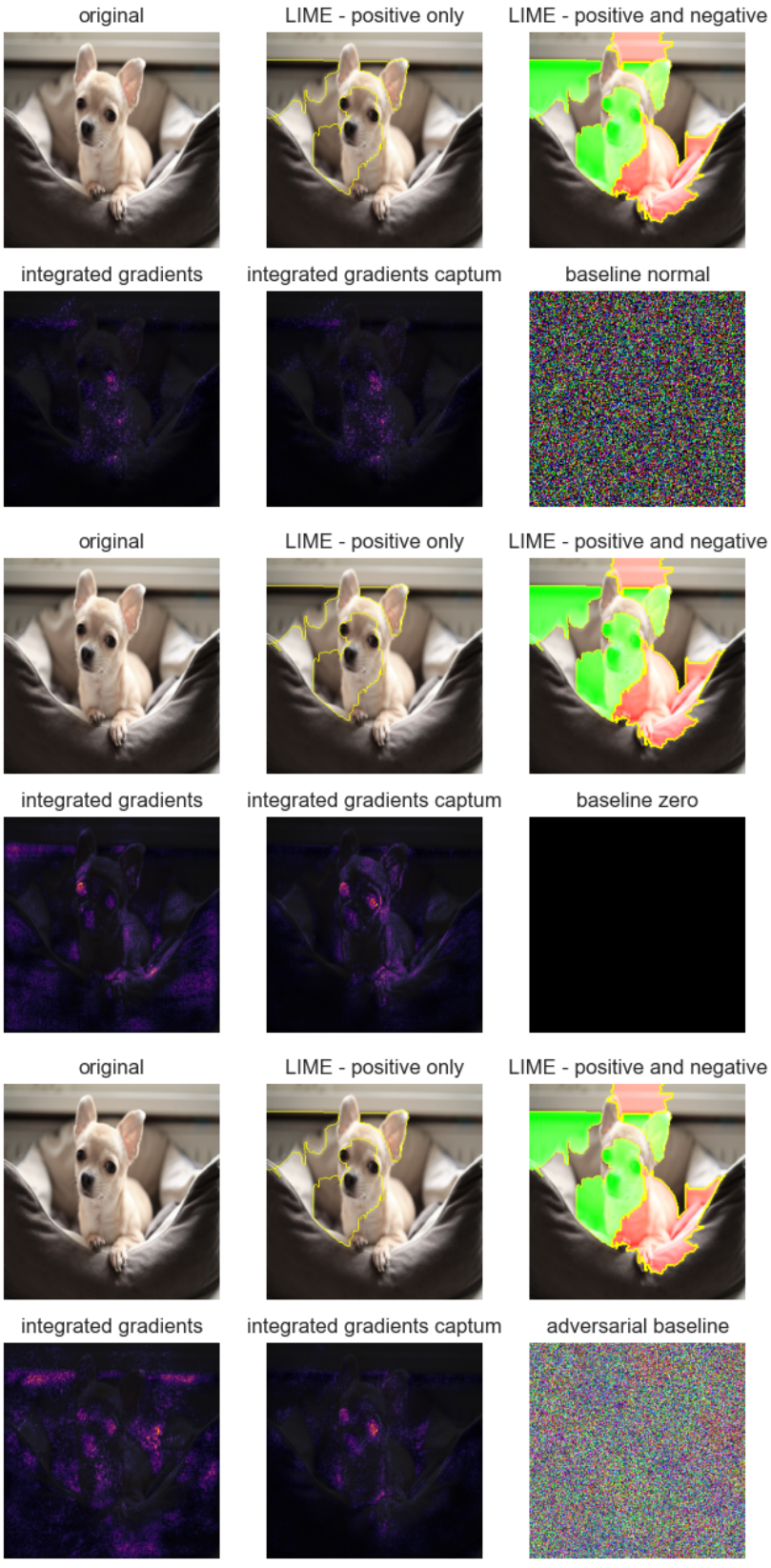
The plots for the two Integrated Gradient methods exhibit a comparable distribution and concentration of gradients, indicating a similarity in their functionality. In contrast, the LIME method displays a slightly elevated concentration of pixels with a value of 1, warranting further examination by inspecting the output mask.

7.5 Analysis 2: correlation plots

For this analysis, we created scatter plots in pairs to visualize the correlation between attribution masks.



Here, the comparison between the two Integrated Gradients methods reveals a tendency for them to align along the same line, indicating a linear relationship between the two approaches. However, when assessing the correlation between LIME and both Integrated Gradients methods, a degree of confusion becomes apparent. Specifically, all methods concur that the eye of the chihuahua is an informative feature, but LIME designates the body of the chihuahua as -1, signifying its exclusion from the chihuahua. This confusion is reflected in the scatter plots, where a substantial concentration of points exists along both the ”1” and ”-1” lines, suggesting that these methods identified different informative features.


original
LIME - positive only
LIME - positive and negative
integrated gradients
integrated gradients captum
baseline normal
original
LIME - positive only
LIME - positive and negative
integrated gradients
integrated gradients captum
baseline zero
original
LIME - positive only
LIME - positive and negative
integrated gradients
integrated gradients captum
adversarial baseline