POLITECNICO DI TORINO

Master degree in Computer Engineering - Data science

Master Degree Thesis

Explaining black-box models in the context of image classification.

Supervisors
Prof. Tania Cerquitelli
PhD. Francesco Ventura

Candidate
Elisa Wan
matricola: s241854

Academic year 2018-2019

Summary

Neural networks can be trained, among other things, to classify the content of an image into different categories. In some cases, neural network surpass human precision. Therefore, the interest toward this technology has increased exponentially in the last few years. Although the performances are very high, the interpretability is low. In this thesis, we make contributions to the development of EBANO, an engine that generates transparency reports for black-box models. First, we compared EBANO's feature extraction step with image segmentation techniques and concluded that the method adopted by EBANO created interpretable features, while the other do not. Secondly, we tested how EBANO reacts to adversarial examples. In this case, the features extracted are completely unrelated to the image content. Then, we trained different VGG16 networks to have unwanted biases, and used EBANO to explain their prediction. In those cases, the nPIR index, an indication of how much a feature affects the final prediction, is not as high as we would expect or shows that the network based its decisions on the wrong features. We also, added the Pascal VOC 2012 dataset to the existing list of supported datasets. And finally, we created a website to show the results.

Acknowledgements

I would like to express my gratitude to professor Tania Cerquitelli for introducing me to the project, for her support over the years and precious advice, and PhD Francesco Ventura who has followed my thesis work closely. I particularly appreciate them for giving me this important learning opportunity and making it a smooth and pleasant experience. A big thanks, also goes to my parents who has always supported my studies and for giving me the means to go forward in live and Francesco Maggiolino for helping me overcame my insecurities.

Contents

Chapter 1

Introduction

In computer science, the field of artificial intelligence is getting more and more popular, companies all around the world are claiming the use of it to improve the perception of their brand. Among all artificial intelligence techniques, the category of artificial neural networks are gaining attention for its capability to adapt to a vast variety of industries, spanning from health-care to transportation and security. In the last years, dramatic improvements were made in this field thanks to technological advancements, creation of large datasets and research on neural networks architecture. This rapid progress with main focus on getting networks more accurate and more efficient has left one important aspect behind, transparency. Taking important decisions based on a black-box model that have been evaluated only on the base of one accuracy value computed on a supposedly representative sample of the real world, is highly risky. Form here, derives the need for an interpretable explanation of the underlying process.

Convolutional neural networks address the image classification problem, which consists of assigning, from a set of available labels, the one the best describes the image content.

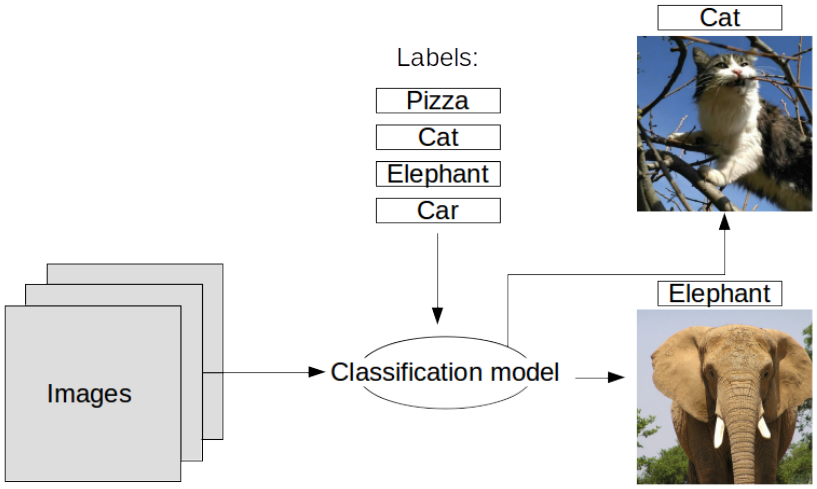


Figure 1.1. Image classification problem consist in assigning to images a representative label describing the depicted subject.

Chapter 2

State of the art

2.1 Convolutional Neural Networks

Back in 1943, the concept of Neural Networks were introduced in [11] by Warren McCulloch and Walter Pitt to describe how biological neurons work.

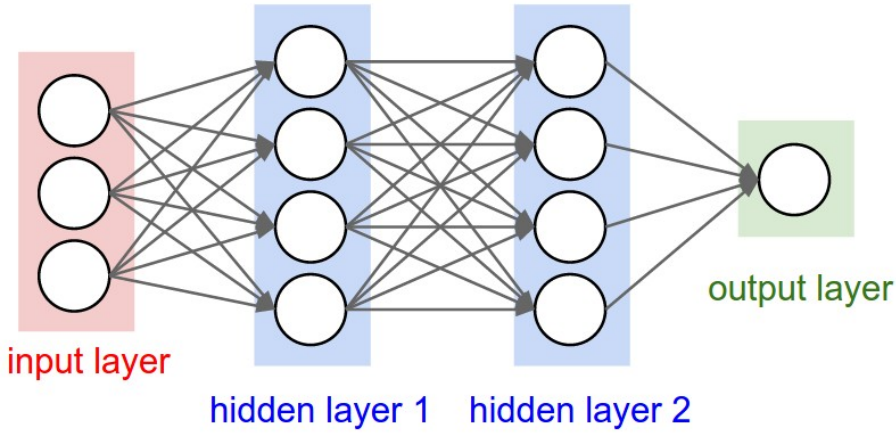


Figure 2.1. General neural network architecture with an input layer, several hidden layers and an output layer.

Later, the discoveries on the visual cortex of mammals and the introduction of the back-propagation algorithm for neural network training gave birth to Convolutional Neural Networks (CNN or ConvNet) in 1989 by LeCun [12], that were specifically designed to deal with images,

although, there are other applications, such as time series analysis and natural language processing.

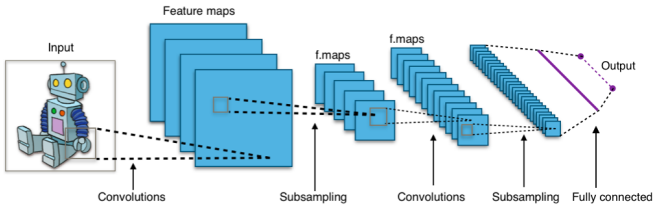


Figure 2.2. Typical ConvNet architecture.

What characterize ConvNets is the usage of a mathematical linear operation widely used in Computer Vision to extract and manipulate images, called "Convolution". The mathematical formula of convolution is:

$$g(t) = \int x(a)w(t-a)da$$

But the 2 dimensional discrete version used for convolution operation on images takes this form:

$$g(x,y) = (\omega * f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t))$$

Original image

1	3	3	2	2
1	1	0	2	2
1	1	0	2	3
0	0	1	2	3
1	0	2	2	1

*

3x3 filter

0	1	-1
1	1	-1
-1	0	0

=

Output image

1	-3	-2
3	-5	-7
-2	-5	-6

Figure 2.3. Convolution between a 6x6 image with a 3x3 filter with stride 1.

2.1.1 Vgg

VGG convolutional neural networks [13] are the result of a study of very deep networks (11 to 19 layers) impact on accuracy. Previously to this work, networks were wider, with convolutional filters of 7x7, but shorter. The general VGG architecture has the following characteristics:

- Input image size of 224x224x3 pixels;
- Convolutional layers with filters of small receptive field: 3x3, 1x1 and stride of 1;
- Stacks of up to 4 convolutional layer interleaved with max-pooling layers;
- Three fully connected layers at the end;
- All layers use ReLU activation function.

Different configurations can be defined following those ground rules (plus others, I have not mentioned here), VGG-16 and VGG-19 are the ones that performed best on ImageNet dataset. From this family of networks, VGG-16 was the chosen one for most of the experiments presented here.

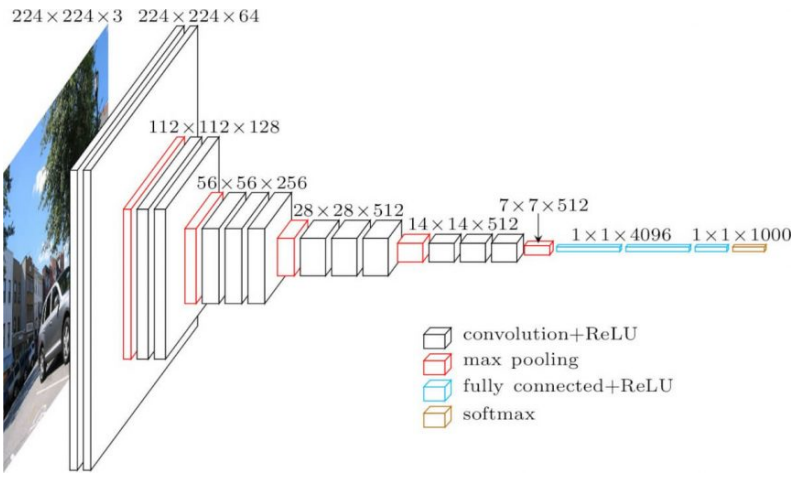


Figure 2.4. VGG16 architecture.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2.5. VGG family network configuration. Image credits to Simonyan and Zisserman, the original authors of the VGG paper.

2.1.2 Inception

Inception [14] is a family of networks created in an attempt to improve the architecture in terms of performance while keeping its computational cost limited and adding the capability of handling images depicting objects at different level of vicinity.

GoogleLeNet, a particular incarnation of the Inception architecture, won the ImageNet Large-Scale Visual Recognition Challenge in 2014. Inception architecture has the following characteristics:

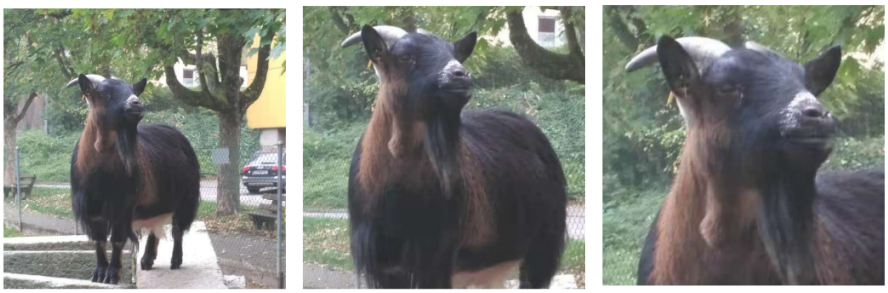


Figure 2.6. The same image at different levels of vicinity

- **Inception modules** are stacked on top of each other. These modules perform multiple convolutions at the same level with different filter size, the result of those are then concatenated to form the input of the next stage.
- **Auxiliary classifiers** connected to intermediate layers to increase the loss value propagated to early layers. These smaller classifiers are used only during training and ignored during prediction.

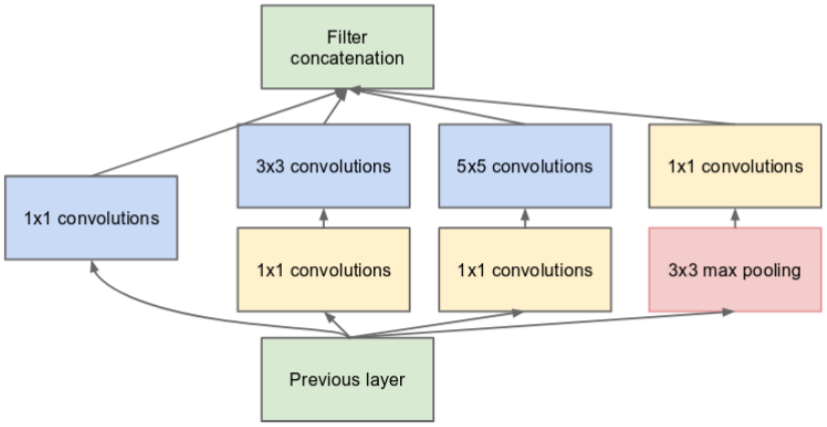


Figure 2.7. Inception module.

In addiction to VGG-16, InceptionV3 and InceptionResNetV2 have been chosen for performing experiments, as well.

2.2 Transparency techniques

In this section, we will explore several techniques that have been proposed in the last few years for explaining black-box algorithms for image classification tasks.

- **Lime**;
- **Grad-CAM**;
- **RISE**.

2.2.1 Lime

LIME, Local Interpretable Model-agnostic Explanations, [2] gives an explanation to predictions of a black-box model by learning a simple linear interpretable model around the locality of the prediction. LIME addressed both text classification and image classification problem, though the text classification problem is ignored here.

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 2.8. Image classification explanation made with LIME using Inception neural network. From left to right: original image, LIME explanation for label *Electric guitar*, label *Acoustic guitar* and label *Labrador*. Areas with positive influence toward the target label are shown. Image from [2]

Let *ConvNet* be the convolutional neural network under examination, *Img* be the image to predict, *RidgeReg* be the ridge regression model used for explaining the prediction. LIME extracts a set of features from *Img* by performing image segmentation with a quickshift algorithm. Those segments are used for creating new images by randomly removing segments from *Img*, let's call them *NeighboorhoodImgs*.

Ones all the *RidgeReg* models have been trained (one for each selected label), the final result is simply the list of features with their corresponding regression coefficient, that can be used for creating images showing only segments with positive coefficient, or only those with negative coefficient or both.

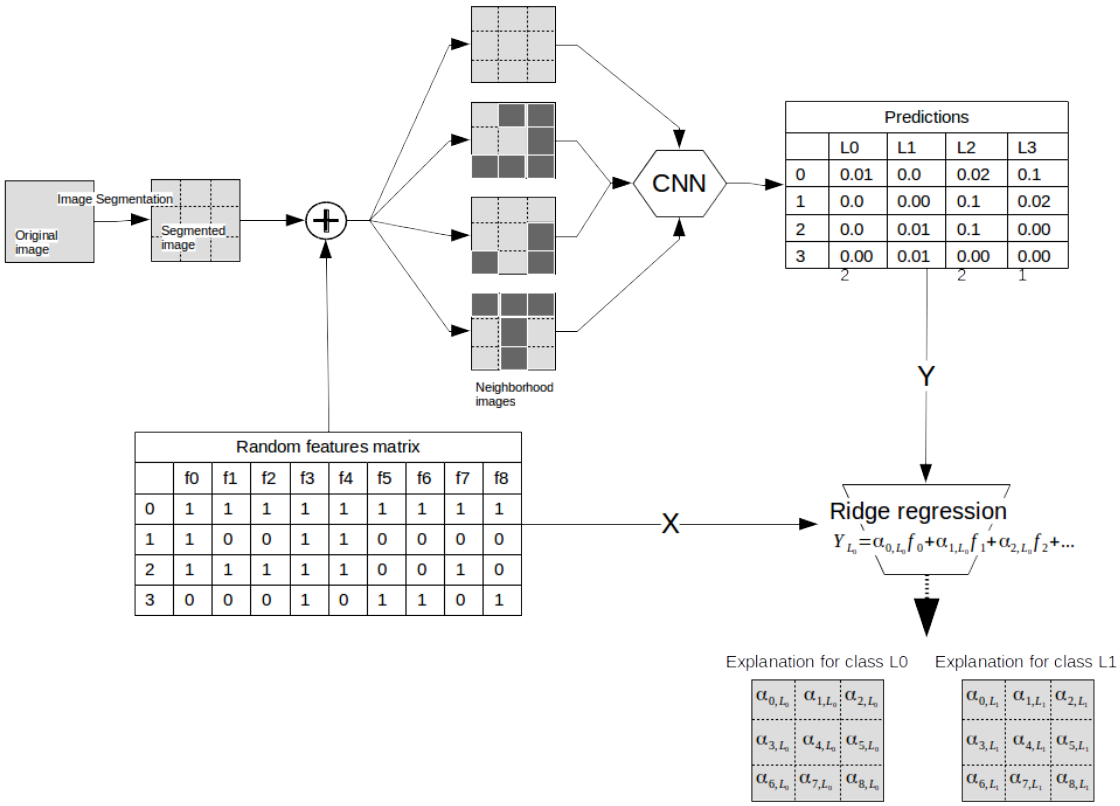


Figure 2.9. LIME overview: the original image is segmented and new images are created from them, each of these new images are fed into the CNN and predicted, finally a linear model is trained and the visual explanation is created according to the linear coefficients.

Figure 2.10. LIME explanation for class "Black Bear": green areas contribute positively to the class, while red areas contribute negatively.

2.2.2 Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) is the visual explanation proposed by researchers from Virginia Tech and Georgia Institute of Technology in [10]. Grad-CAM base its visual explanation on the gradient information going into the last convolutional layer, before the spacial information is lost in the fully-connected layers. In addiction to Grad-CAM, in the same paper [10], they propose an alternative called "Guided Grad-CAM" that combine Guided Backpropagation with Grad-CAM for more detailed information.

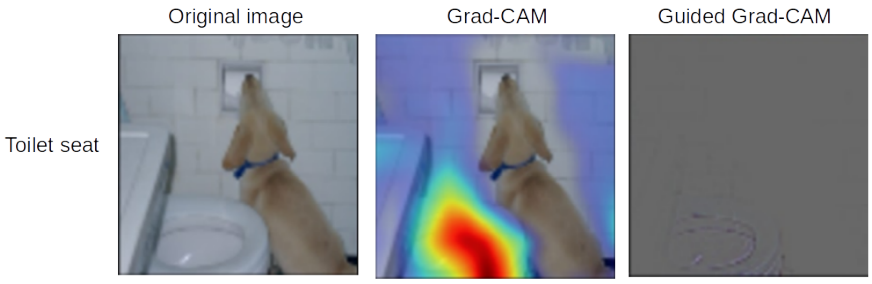


Figure 2.11. Grad-CAM and Guided Grad-CAM visual explanation for the class "Toilet seat".

GRAD-CAM intuition is based on previous work that have demonstrated that deeper convolutional layers of a CNN capture high-level visual structures, whereas shallow layers captures low-level structures (i.e. dots, lines, ...). For this insight, they expect the last convolutional layer to have the information needed for the creation the desired visual explanation.

2.2.3 RISE

RISE, an approach developed at the Boston University, creates a saliency map that for each pixel tells how important it is for the network prediction. For this approach, the classification model is completely black box, meaning that we do not have access to its internals, unlike other approaches. All they can consider is the input and the output.

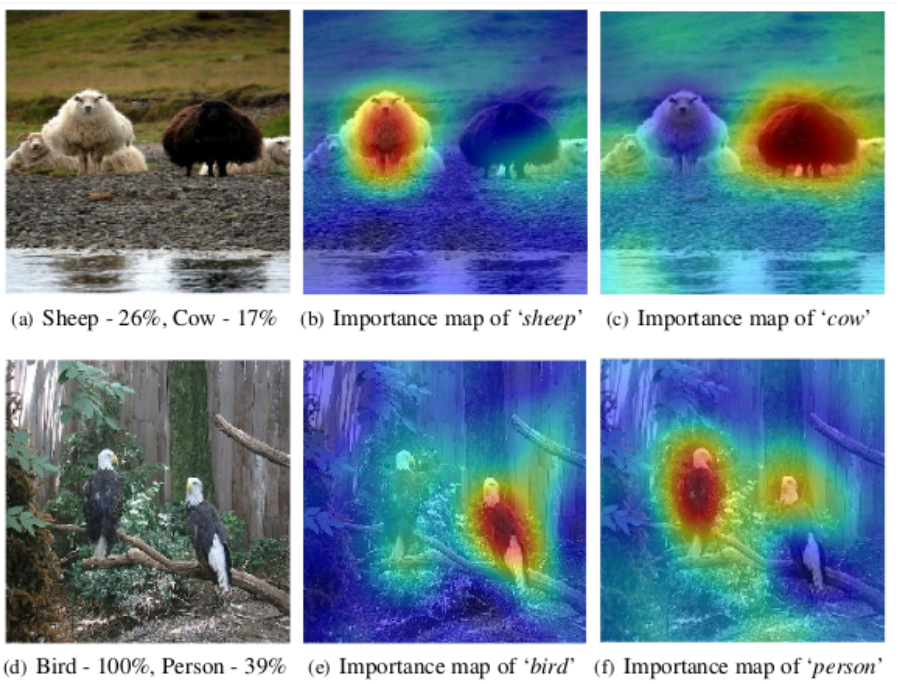


Figure 2.12. RISE heat map for two different images: Original image on the left.

To estimate the importance of each pixel, they randomly mask parts of the input image to get different classification outputs. Then, the saliency map generated as a linear combination of the randomly generated occlusion masks multiplied by the predicted probability, as shown in figure 2.10.

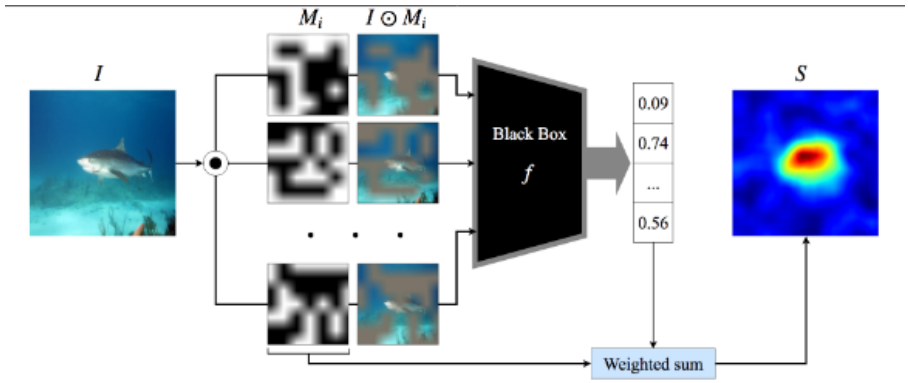


Figure 2.13. RISE overview.

Chapter 3

Used technologies

Tools and libraries used for this work are the following:

- **Python** the most used programming language among data scientists.
- **PyCharm** an integrated development environment (IDE) for Python code.
- **Jupyter notebook** a web-application for creating documents with text, code (eg. Python) and its output.
- **Keras** a high-level deep learning library for Python that runs on TensorFlow, Theano or CNTK.
- **OpenCV** a computer vision library.
- **Dash** a Python framework based on Plotly.js, React and Flask for building web applications with interactive graphs.

Chapter 4

Approach

4.1 Ebano

EBANO (Explaining BlAck-box mOdel) is the starting point of this work. It is an engine, at its early stage, which final goal is to provide to general users, the tools for a better understanding, through simple and interpretable explanations, of deep convolutional neural networks. Given a pre-trained CNN and an image, EBANO partitions the latter and outputs a transparency report with indication of how each partition influenced the classification outcome.

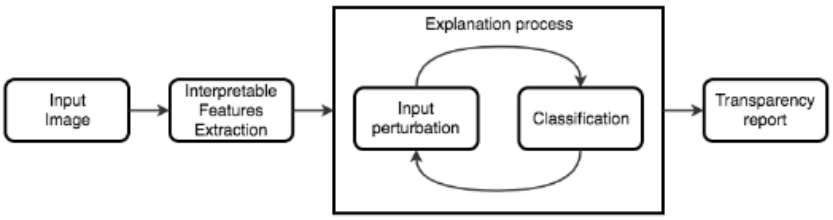


Figure 4.1. Ebano process overview.

When dealing with unstructured data, such as images, the definition of what constitutes a feature is not straightforward, the size or the time-stamp is not related to the content, a single pixel is not representative of an entire image, but a group of correlated pixels could

contain enough visual information about an image's content. The strategy adopted by EBANO to divide the image into subsets of correlated pixels is to perform a SDS (Simultaneous Detection and Segmentation) analysis based on hypercolumns and clustering. Once, EBANO has identified a set of features with the above technique, it creates a new set of images each having one feature perturbed and classifies each of these with the CNN under examination to compare the results with of the classification result of the original (not-perturbed) image. From the differences between the classification of the perturbed images and the original one, two indexes called nPIR and nPIRP are produced.

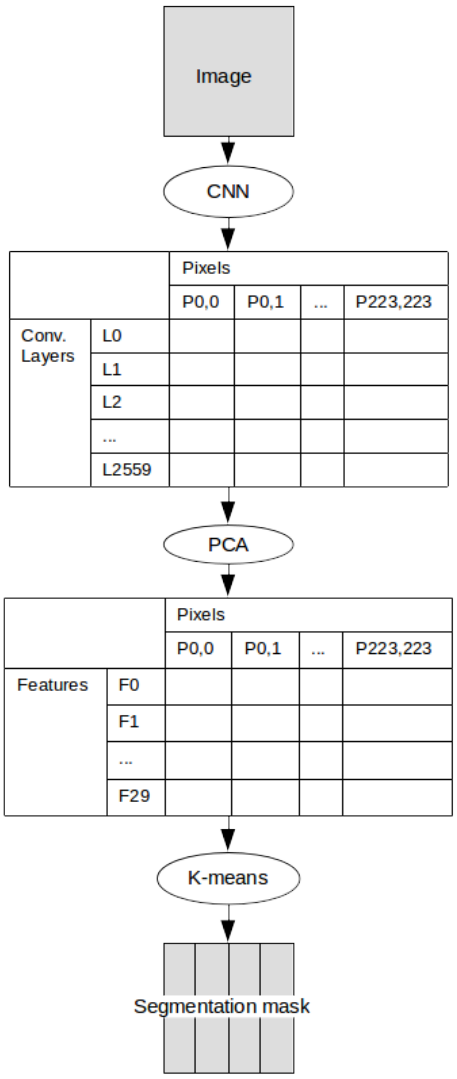


Figure 4.2. EBANO feature extraction step: extract hypercolumns for each pixels from the last 10 convolutional layers, then reduce dimentionality to 30 with principal component analysis and perform k-means to find clusters of similar pixels.

4.2 Image segmentation

In the previous section, the problem of extracting features from an image and the solution provided by EBANO have been introduced, but in the Computer Vision's literature, there are already a set of well-known algorithms that address this problem, called Image Segmentation. In this section, some of those algorithms, offered by OpenCV, a widely used computer vision library, will be examined and compared.

- Flood fill;
- Watershed;
- GrabCuts;
- Meanshift.

Many other algorithms are available, but for the scope of this work, we decided to try just a few of them.

4.2.1 Flood Fill

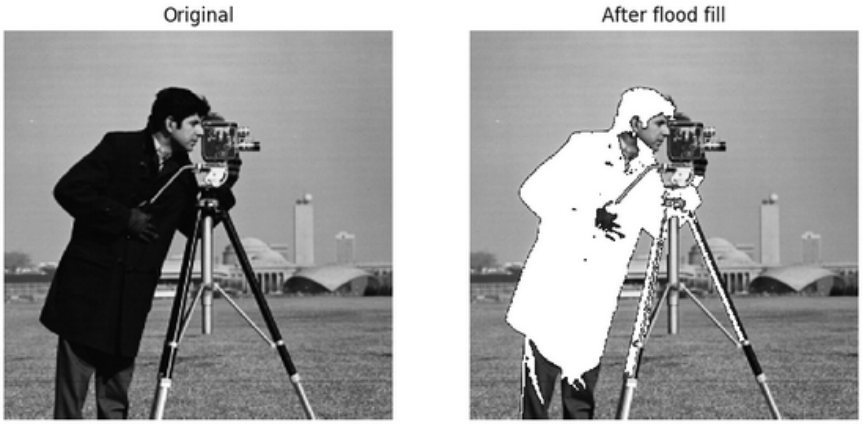


Figure 4.3. Image segmentation with flood fill algorithm. Source: https://scikit-image.org

Flood fill is a simple operation for segmenting one area out from the rest of the image by only using intensity information. Starting from a point called "seed" with an intensity of I_0, neighboring points are included to the segment if their intensity value lies within a certain range $[I_0 - lowDiff, I_0 + highDiff]$. The result of this operation is one continuous region containing the seed.

This algorithm presents several characteristics that makes it unusable in our scope: firstly, it needs an input, usually given by a human, the so called "seed"; secondly, it is able to identify only one segment. The latter, would be easily overcome by running the algorithm several times with different seeds, but the first problem of positioning the seed still remains. For this reason, no further investigation has been done in this direction.

4.2.2 Watershed Algorithm

The watershed algorithm interprets images as topographic surfaces of the image intensity gradient with "mountains" and "valleys". Valleys are flooded from their minima until waters coming from different sources merges. This technique works better when it starts from human selected points, opposed to starting from every minimum point which produces over-segmentation. Variation of the traditional Watershed algorithm exists, in particular the hierarchical approach can be used to overcome the over-segmentation problem, in addiction, the P algorithm variation of the hierarchical watershed segmentation does a really good job [17]. Unfortunately, only the traditional watershed algorithm is available in OpenCV.

We applied a strategy proposed in [18] that applies the watershed algorithm with input markers dynamically found through a series of morphological transformations and thresholding. As shown in figure 4.2, the results are quite good since reasonable segments have been found, therefore we will explore in section 5.1 the possibility of substituting the current EBANO's feature extraction step with this watershed implementation.

These are the steps performed:

- **Otsu thresholding** on the gray scale version of the image, this will hopefully extract the foreground from the background.
- **Opening**, a morphological operation to remove noise.
- **Dilation**, a morphological operation to dilate the background, this will increase the probability that all background has been extracted.
- **Distance transform** returns the distance from each foreground point to the closest background point.
- **Thresholding** on the distance transform to get the points that are for sure part of the foreground object
- **Connected components** on a binary image with all foreground points at 1.

- **Watershed** segmentation technique using connected components as markers.

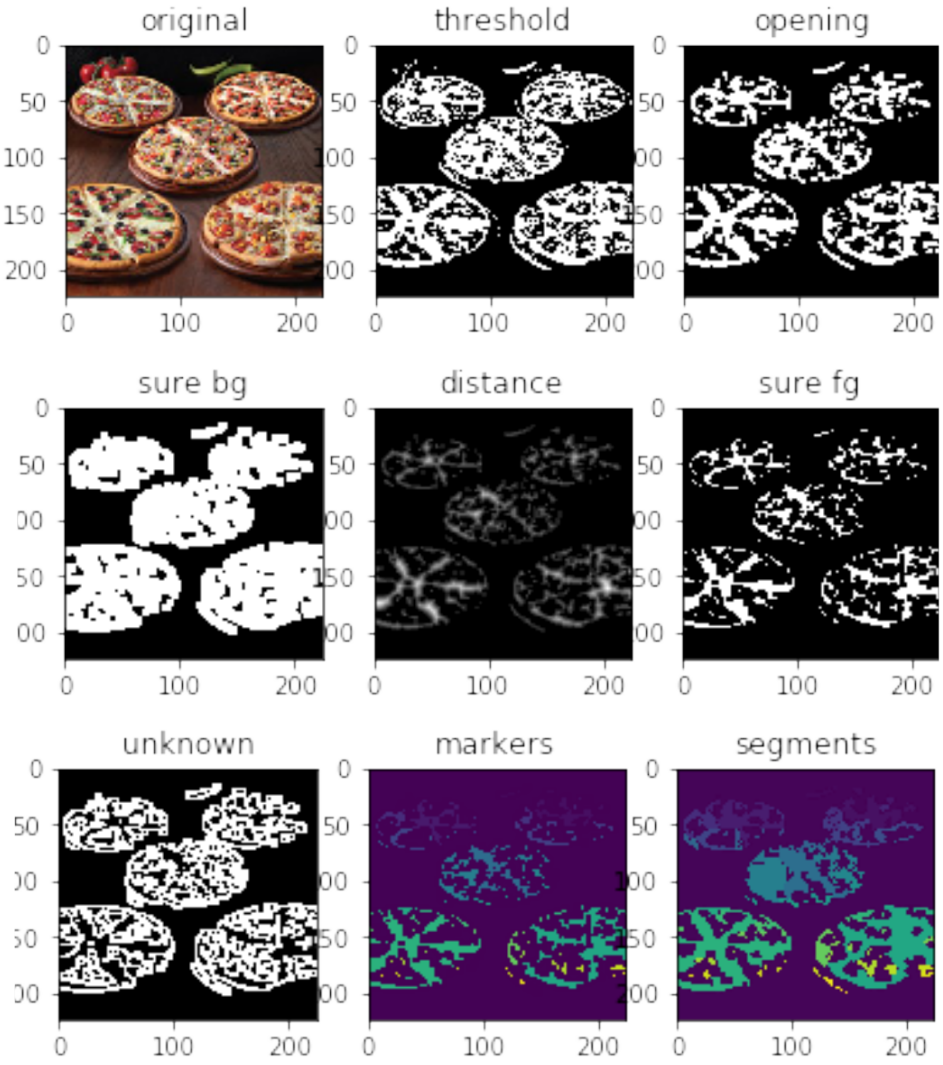


Figure 4.4. Steps of image segmentation with watershed algorithm on image of pizza.

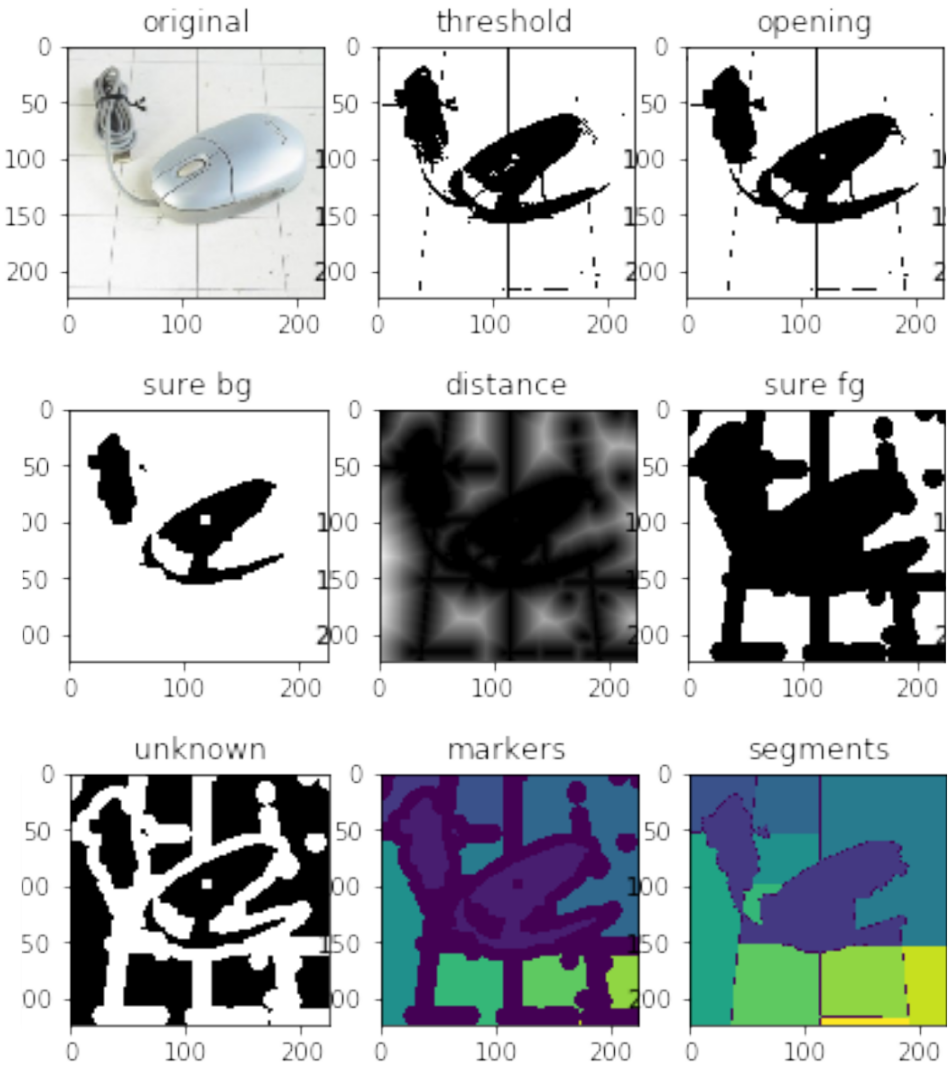


Figure 4.5. Steps of image segmentation with watershed algorithm on image of mouse.

Figure 4.6. Image segmentation with watershed algorithm.

4.2.3 Grabcuts

Grabcuts algorithm [19], is another image segmentation method, but it focuses on extracting the foreground from an image, plus it needs human input to draw a rectangle around the foreground object to be extracted.
GrabCuts works as follow: the input rectangle is used to mark everything outside as background and everything inside as unknown, background and foreground are modeled with a GMM (Gaussian Mixture Model) that creates a pixel distribution based on color information and labels them as probable background or probable foreground depending on their relation to the already labeled pixels. At the end, a graph is created with each node representing a pixel and edge weights the similarity between pixels. The foreground and the background are separated at the and using a mincut algorithm.

As for the Food Fill algorithm, GrabCuts needs human input and extracts only one segment which makes this algorithm not a good choice for the feature extraction phase of EBANO.

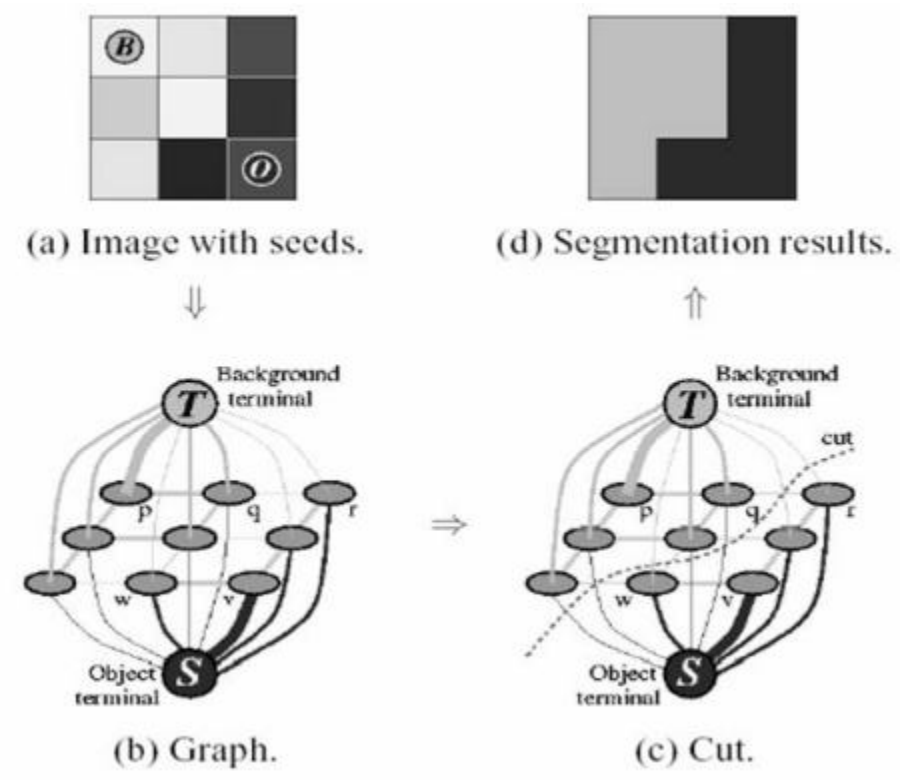


Figure 4.7. Grabcuts segmentation algorithm.

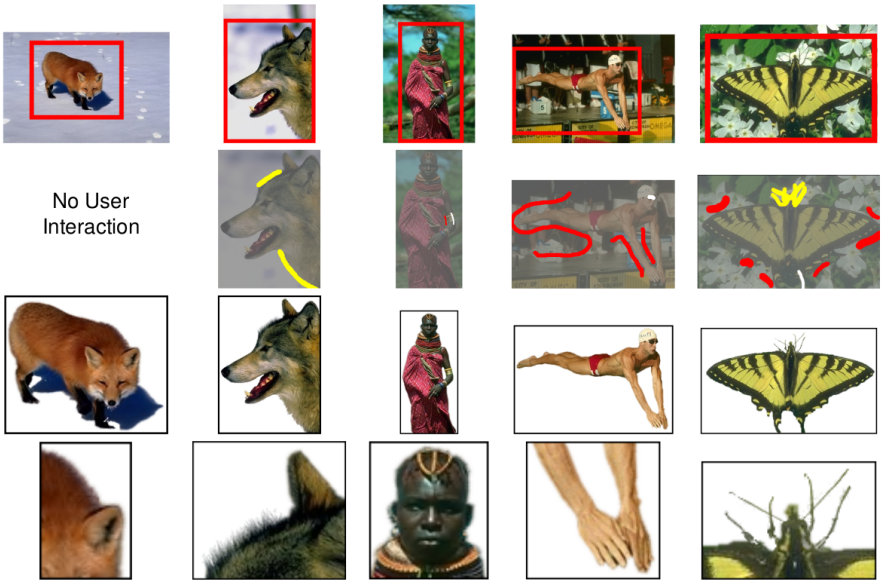


Figure 4.8. Image segmentation results using GrabCuts. Image taken from original paper [19].

4.2.4 Mean-Shift Segmentation

Meanshift, is a clustering based segmentation method that given a spatial window size s and a color window size c, iterates over each pixel and create clusters of similar pixels. The difference between meanshift for image segmentation and data clustering is that with image we have to consider not only the spatial distribution but also the color distribution, hence the need of two different window size.

For each pixel, a window of size s is created and the mean value of the pixels within the window is computed, then the window shift to the mean value just calculated, this process is repeated until convergence. Pixels with windows ending up at around the same point, will belong to the same segment.

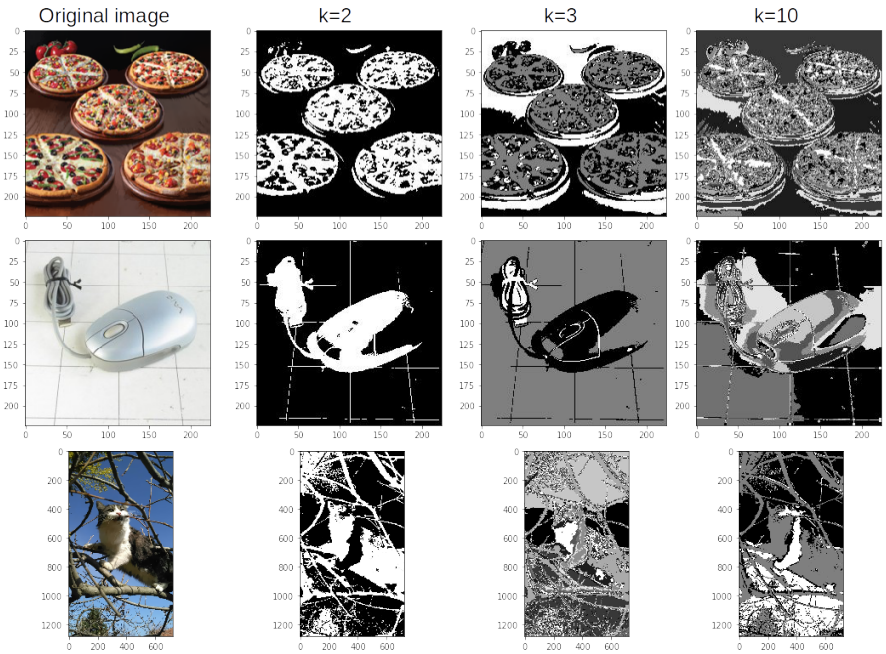


Figure 4.9. Image segmentation with meanshift algorithm at different values of k, the number of clusters.

4.3 Adversarial images

Convolutional neural networks, as well as lineal models, are vulnerable to adversarial examples, tailored inputs made explicitly for misleading the classifier [9]. Adversarial examples for ConvNets can be created by taking a normal image and applying very small pixel variation, unnoticeable to the human eye. Naturally, this is one major concern related the the robustness and trustability of the model.

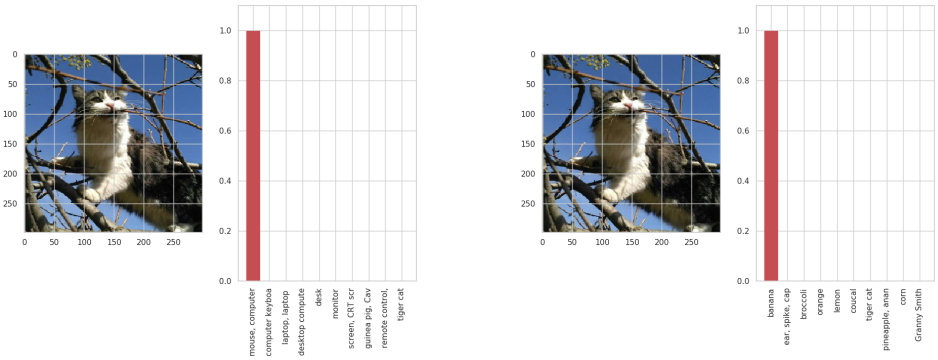


Figure 4.10. Adversarial example: a cat image modified to induce the network to classify them as mouse and as banana

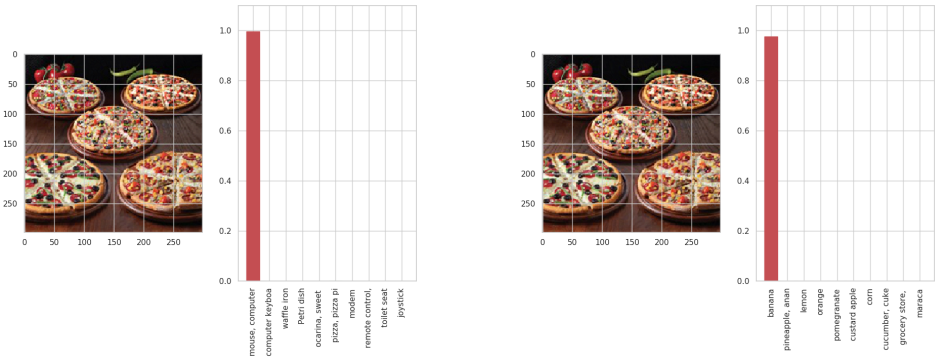


Figure 4.11. Adversarial example: a pizza image modified to induce the network to classify them as mouse and as banana

Is EBANO robust to adversarial examples? To answer to this question, we created some adversarial examples and asked EBANO to explain the predicted output. In figure 4.4 and 4.5 it's shown how two images that seem identical to the human eyes are completely different to the neural network, on the left images are classified as 'mouse' and on the right as 'banana', both with really high confidence.

The interpretability report produced by EBANO is not able to correctly classify the adversarial examples, but the produced result is very different from what we are used to.

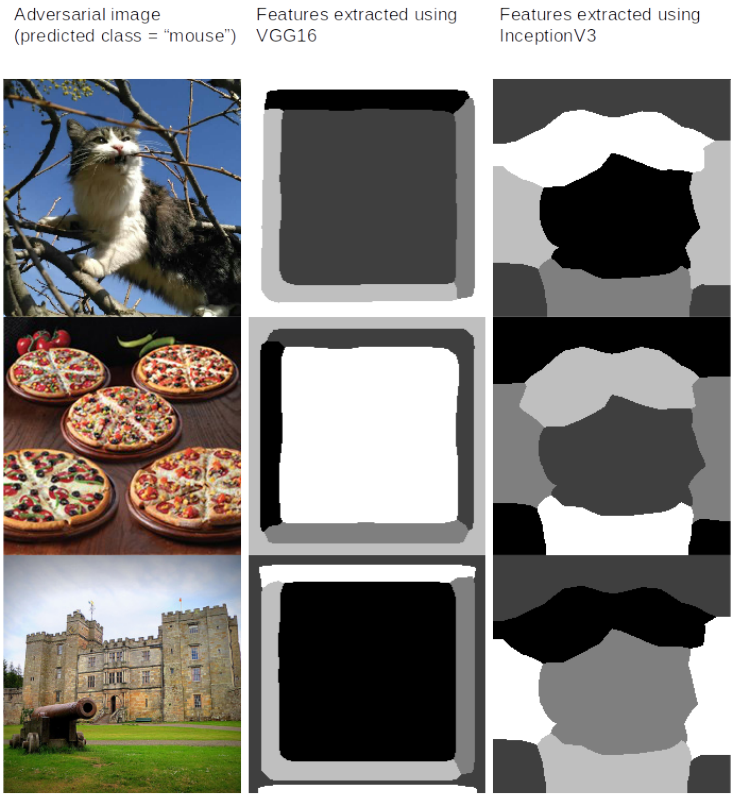


Figure 4.12. Features extracted by EBANO on adversarial examples are unrelated to the image content, but depend on the neural network used for prediction.

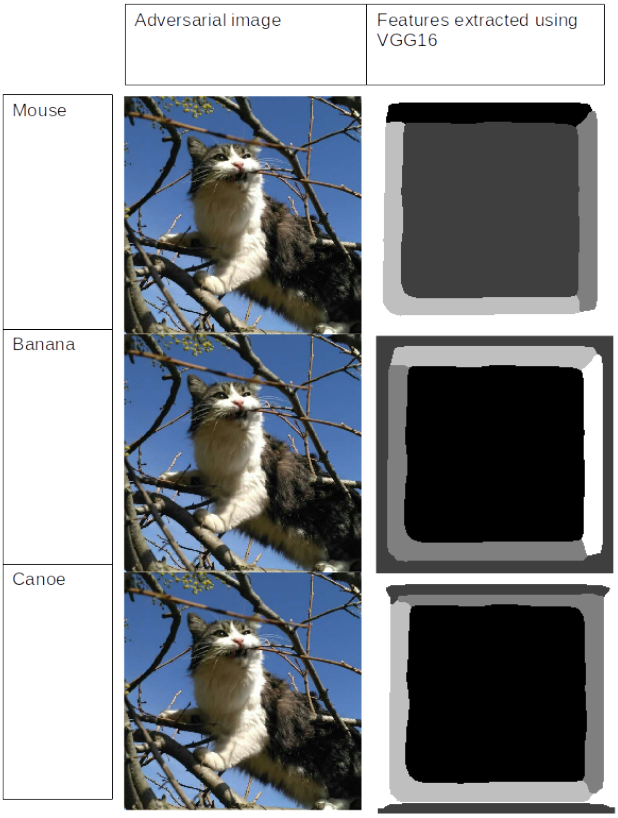


Figure 4.13. Features extracted by EBANO on adversarial examples are unrelated to the predicted class.

In figure 4.6 we show the results of feeding EBANO different adversarial images, all miss-classified as "mouse" by VGG16 and InceptionV3 networks. Extracted features from the VGG16 network are almost identical, independently from the input image, and the same is true for InceptionV3. Moreover, also adversarial images miss-classified as "mouse", "banana" and "canoe" all produce the same set of features regardless of the predicted class, as shown in Figure 4.7. We conclude that, EBANO is not robust to adversarial examples, but the shape of the extracted feature could be used for raising suspicions on the nature of the input image. However, the conclusion has been drawn only based on adversarial examples created using the same method.

4.4 Biased model

Black box classification models might have hidden biases that are hard to discover. However, there are some common mistakes that we can reproduce and see how EBANO will explain the classification and whether it is possible to use EBANO to evaluate a model's quality.

- Training set is unbalanced, not all classes are equally represented;
- Training set does not generalize the real use case.

In order to reproduce those scenarios, we created specific datasets and trained VGG16 models from scratch, without using transfer learning. The reason for not using transfer learning is to remove any information that the network have correctly learned on other datasets. Here, the goal is to verify EBANO's usefulness on detecting cases where the neural network has learned wrong features, to do so, we simulate those situations where the training dataset has high intrinsic biases that will lead to wrong learnings.

The following configuration has been used:

- Output classes: [Airplane, Train];
- Activation function: Softmax
- Optimizer: Adam
- Learning rate: 0.00001
- Loss function: categorical crossentropy

```
VGG_16_____________________________________________________________
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, None, None, 3)     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              525312
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_3 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 1026
=================================================================
Total params: 16,815,426
Trainable params: 16,815,426
Non-trainable params: 0
```

Figure 4.14. VGG16 Convolutional Neural Network architecture

4.4.1 Dataset unbalance

One typical problem is dataset unbalance, meaning that the number of samples varies a lot between classes. For example, if a training set as 1000 samples in class A, but only 10 samples in class B, then the model will likely learn, during training, that all samples belong to class A, with a training accuracy of 0.99.

We trained a convolutional neural network for image classification on a dataset specifically designed to be unbalanced in the number of images per class:

- Airplane: 1000;
- Train: 10.

Using EBANO's transparency reports on models that have been trained on unbalanced dataset will show an inconsistency between the predicted class and the nPIR indexes. In correctly classified images, the predicted class will have a very high level of confidence (around 0.99, in our examples), but the nPIR plot will show that the features containing the object has a very low nPIR value for the correct class and a very high negative value for the other classes. This same behavior has been seen in all analyzed images. As for the images that are wrongfully classified, the situation is the same, the high confidence is not backed by any feature nPIR value. One very good example is given by Figure 5.8, the nPIR values for class 'Plane' are all close to zero, despite this, the image as been classified as 'Plane' with a 0.99 probability.

Figure 4.15. Example of images contained in the unbalanced dataset used for training.

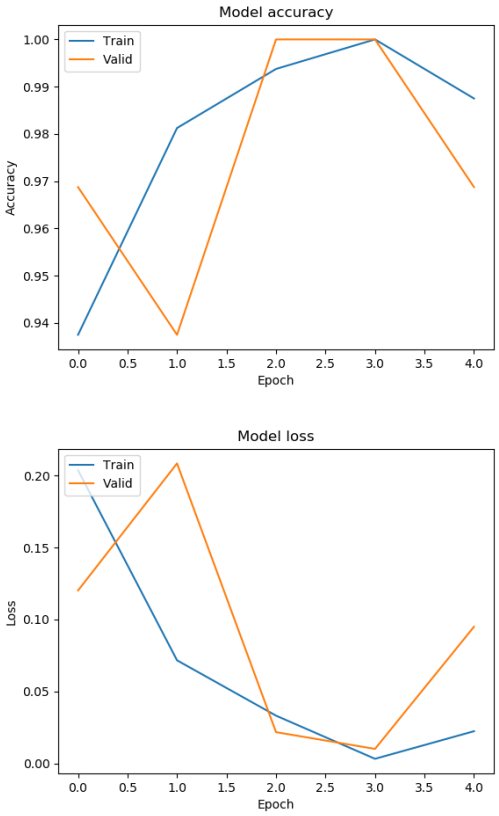


Figure 4.16. Model accuracy and loss

4.4.2 Dataset is not general enough

Neural networks learn by example, iteration after iteration they adjust weights proportionally to the prediction error on the previous iteration. This means, that the training set has to be representative of the data the network will receive in production for it to perform as well as in testing. A VGG16 network has been trained on a dataset that contains 'Airplane' and 'Train' with the particularity that all airplanes are flying in the sky.

This network tends to classify as 'Plane' not by looking for the actual plane in the image, but based on the presence of a blue sky. Such a insight can not be extracted with statistical values like accuracy, sensitivity, etc.

EBANO's transparency report has proved to be able to give useful information for exposing cases where the network has not learned the correct features, even if the classification turns out to be correct.

Figure 4.17. Example of images contained in the dataset used for training a network that has not been able to lean the correct features. An airplane in the sky on the left and a train on the ground.

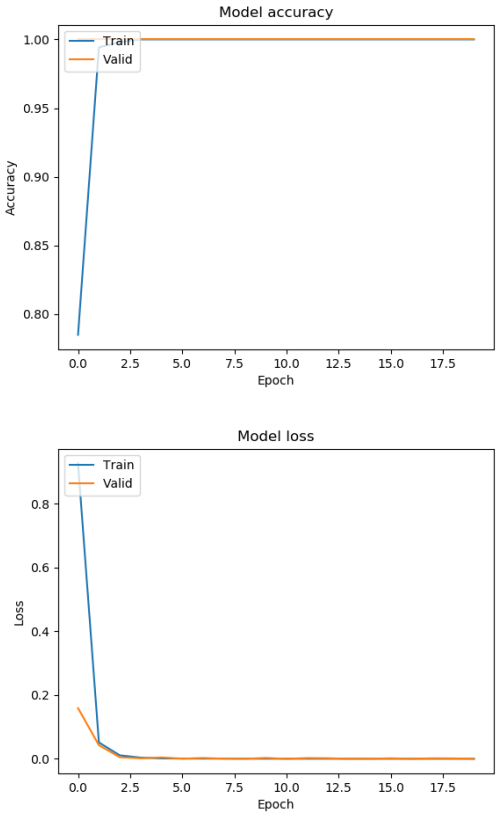


Figure 4.18. Model accuracy and loss.

4.5 Pascal VOC dataset

Pascal VOC data set is a collection of images depicting realistic scenes with objects belonging to 20 different classes:

- Person: person
- Animal: bird, cat, cow, dog, horse, sheep
- Vehicle: airplane, bicycle, boat, bus, car, motorbike, train
- Indoor: bottle, chair, dining table, potted plant, sofa, TV/monitor

The pascal VOC data set is part of a bigger project that has run a series of challenges from 2005 to 2012 providing a benchmark for image recognition/detection and image segmentation to the machine learning community. Adding support for this dataset in addition to ImageNet, COCO and Caltech is just another way of making EBANO more versatile and providing a wider range of images to test.

Figure 4.19. Examples of images contained in the Pascal VOC 2012 dataset.

4.6 Website

The final goal of Ebano is to provide to a general, non-expert user of a convolutional neural network an easy tool for assessing the quality of a network. Looking at that direction, it seemed useful to create a simple website to showcase the results and, hopefully, use it for doing evaluations with human subjects on the interpretability and usefulness of EBANO's transparency report.

The website provides a view of all images available with the possibility to select:

- Model: Vgg_16, Inception_V2, ...
- Dataset from which the image has been taken: Coco, ImageNet, ...
- Class of interest: Pizza, Banana, Elephant,...

By selecting an image, the transparency report of EBANO is shown, containing:

- List of features;
- nPIR-nPIRP graphs;
- Distribution of predictions before and after blurring a feature;
- GRAD-CAM explanation (if available);
- LIME explanation (if available).

A non-expert user should be able to distinguish a good classification from a bad one looking at information provided and a quick definition of those indexes.

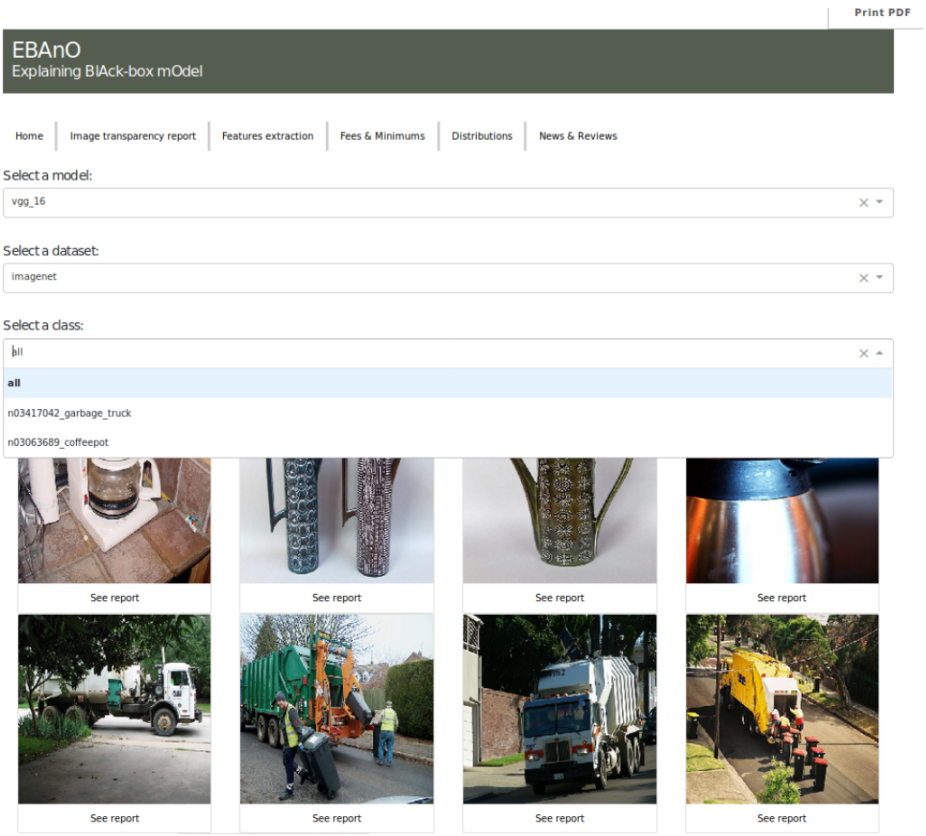


Figure 4.20. Web site home page shows a list of images for which reports are available.

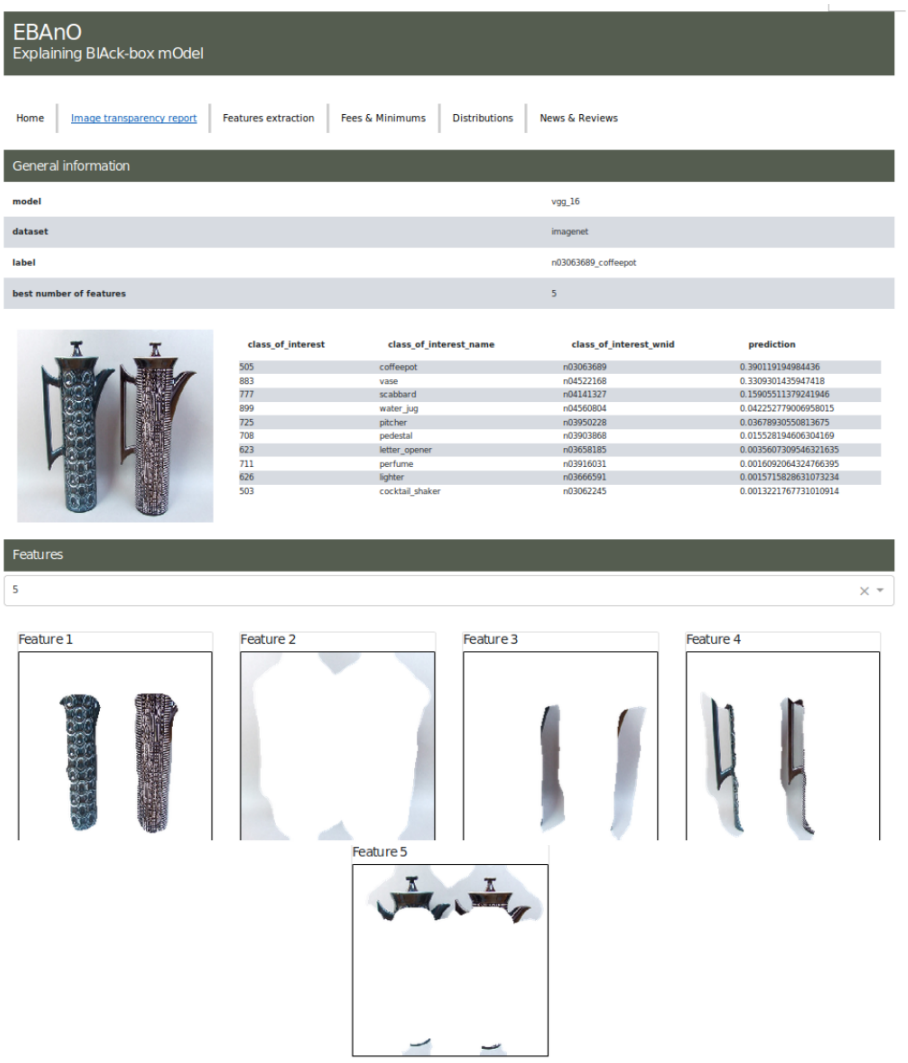


Figure 4.21. Web site report page of coffeepot image. Starting from the top: a list of general information, the image under analysis, top-10 predicted classes, list of all the features extracted by EBANO.

Figure 4.22. nPIR and nPIRP plots shown in the web page for class coffeepot, with possibility to select a different class from a drop down list.

Figure 4.23. For each feature, prediction distribution before and after the image has been perturbed by blurring the feature.

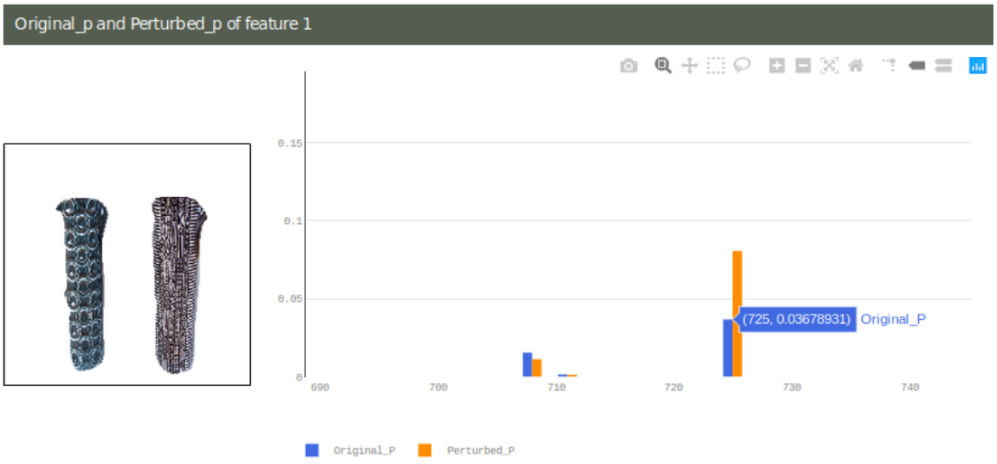


Figure 4.24. Plotly allows interactive interaction with plots: pan, selection, download, zoom and more operations are possible.

Chapter 5

Experimental results

5.1 Feature extraction with image segmentation

In this section, we will explore the possibility of substituting the current EBANO's feature extraction step, based on hypercolumns, with traditional computer vision segmentation methods.

Among the different methods we analyzed in chapter 4, only two were really applicable to our case:

- Meanshift;
- Watershed.

In the following pages, we will compare features extracted by Ebano using hypercolumns, meanshift and watershed. For each image and each method, these are the information presented:

- Original image;
- Neural network used for inference;
- Segmentation method;
- Correct class label;
- Predicted class label;

- Image with extracted features with hypercolumns, meanshift or watershed;
- nPIP and nPIRP plot for correct class:
- nPIP and nPIRP plot for predicted class if different from the correct class otherwise an image containing the feature with highest nPIR value.

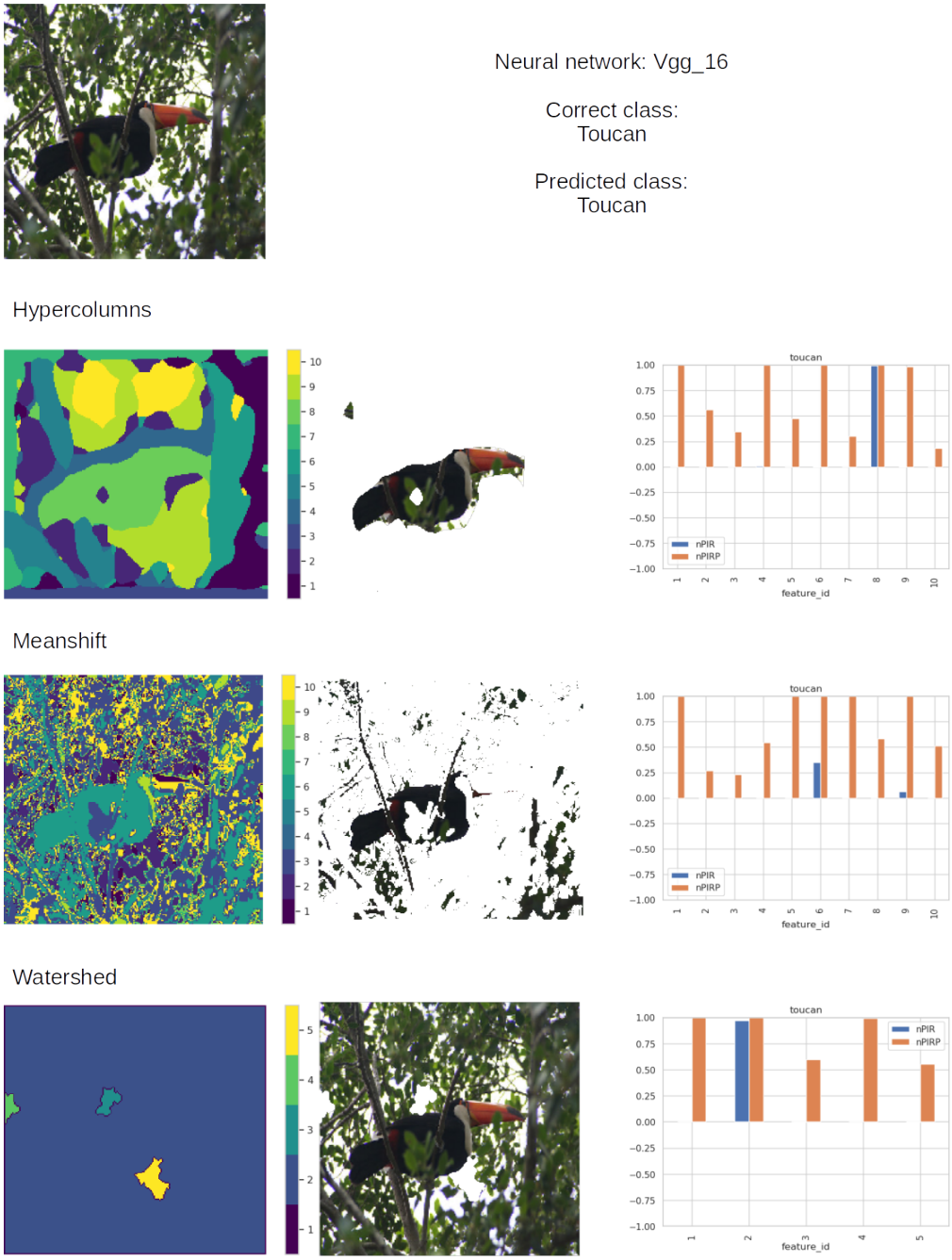


Figure 5.1. Different segmentation methods applied on image of Toucan.

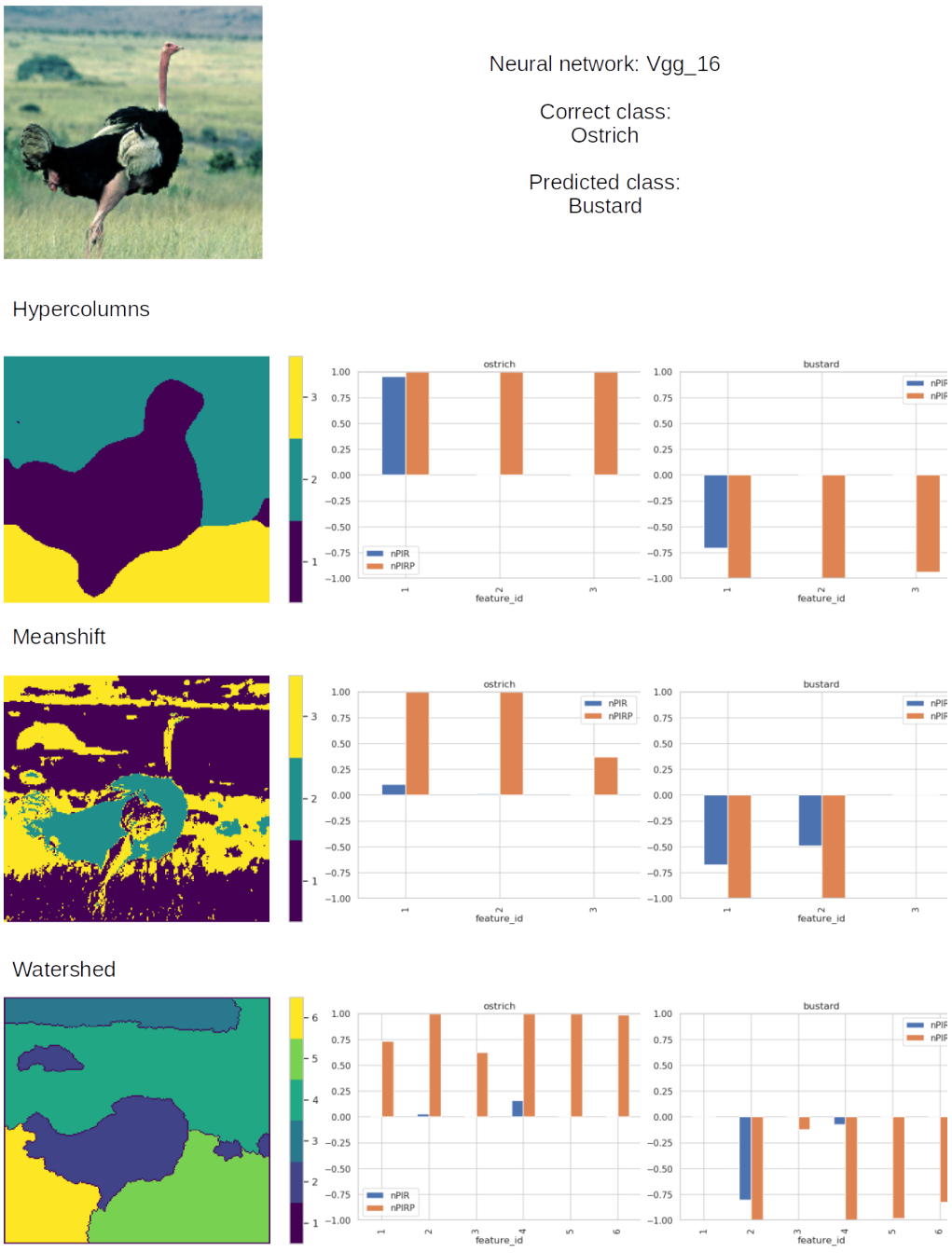


Figure 5.2. Different segmentation methods applied on image of Ostrich.

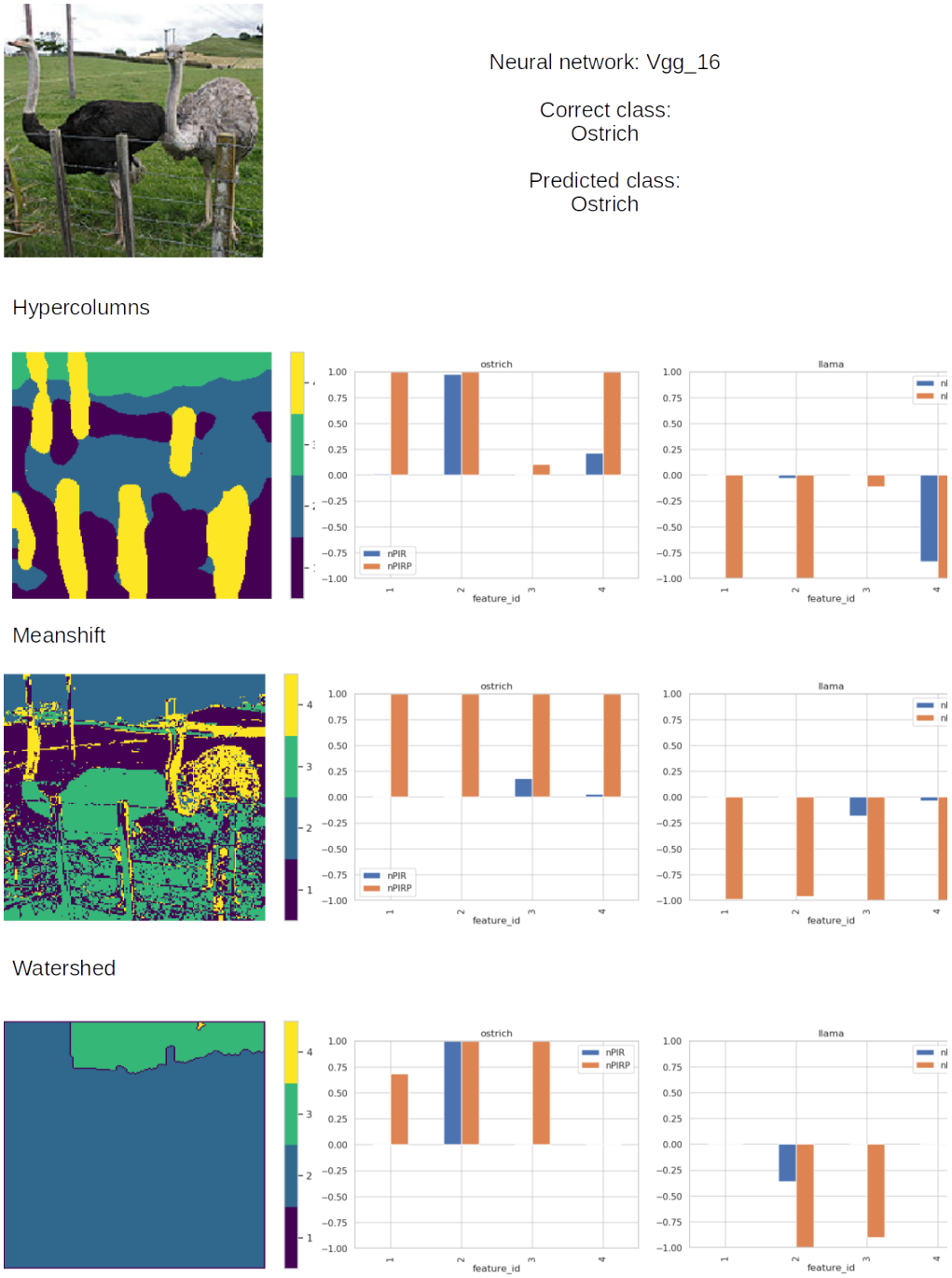


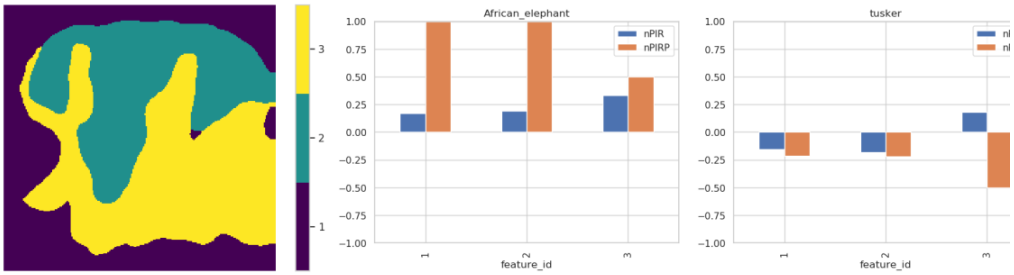
Figure 5.3. Different segmentation methods applied on image of Ostrich.

Neural network: Vgg_16

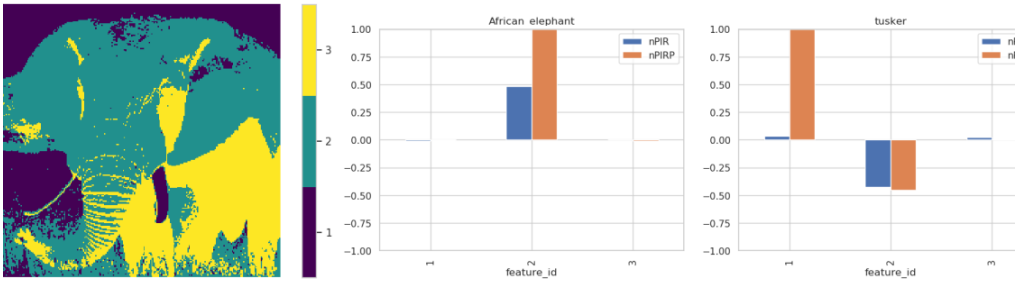
Correct class:
African elephant

Predicted class:
African elephant

Hypercolumns



Meanshift



Watershed

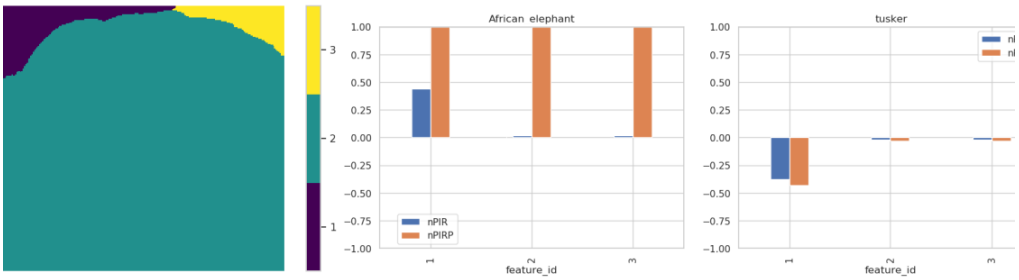


Figure 5.4. Different segmentation methods applied on image of African elephant.

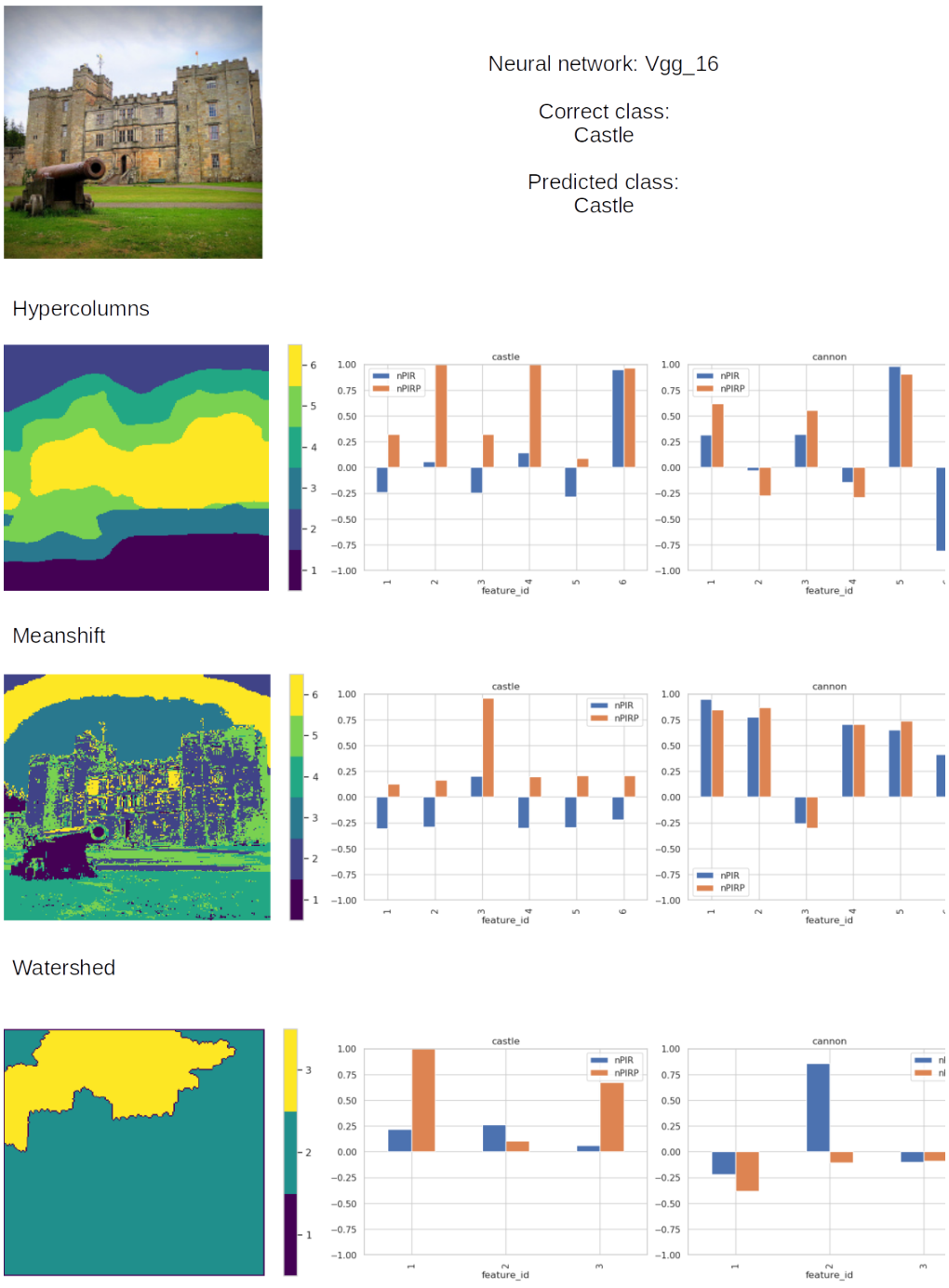


Figure 5.5. Different segmentation methods applied on image of Castle.

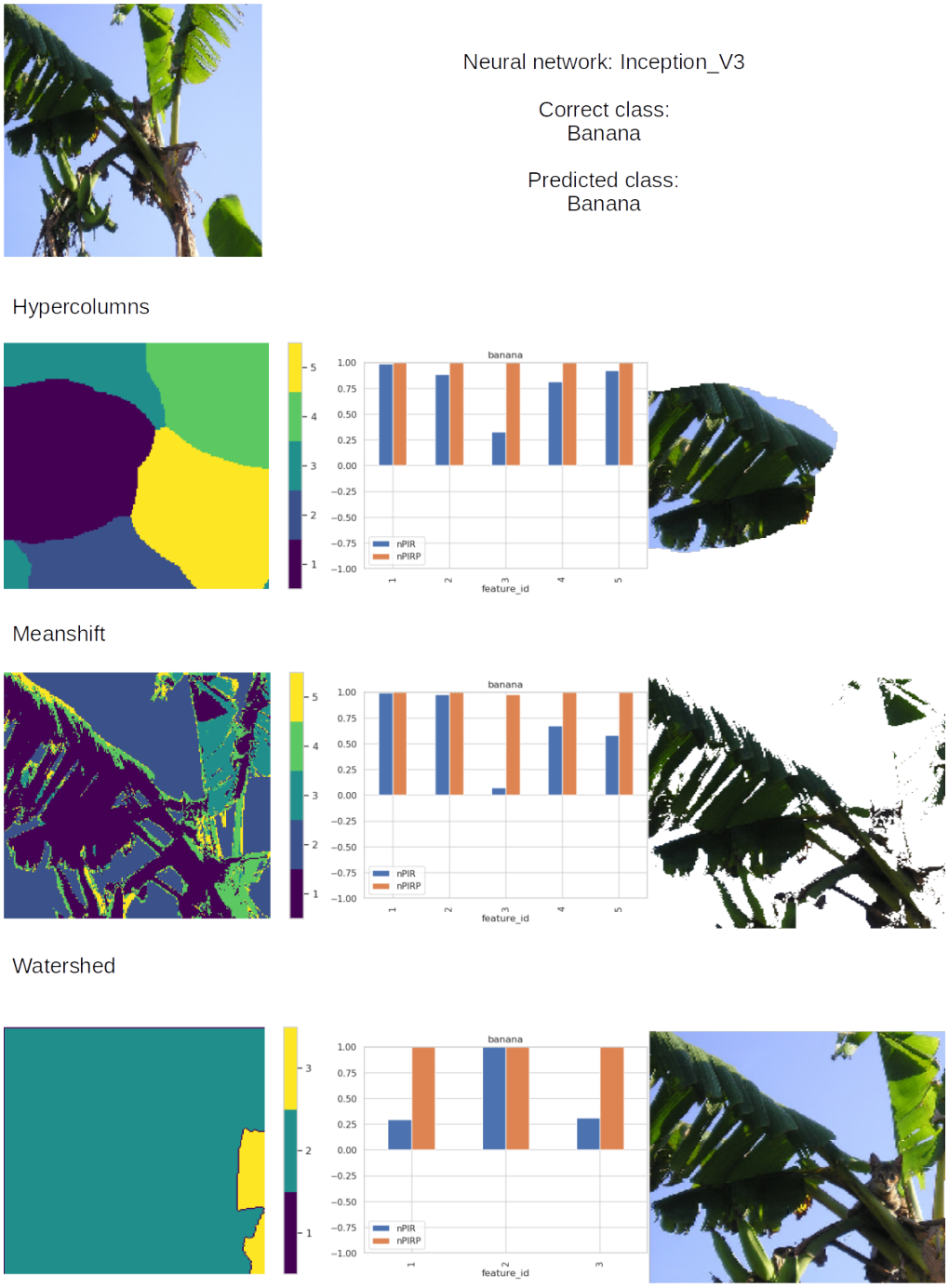


Figure 5.6. Different segmentation methods applied on image of Banana using InceptionV3 network.

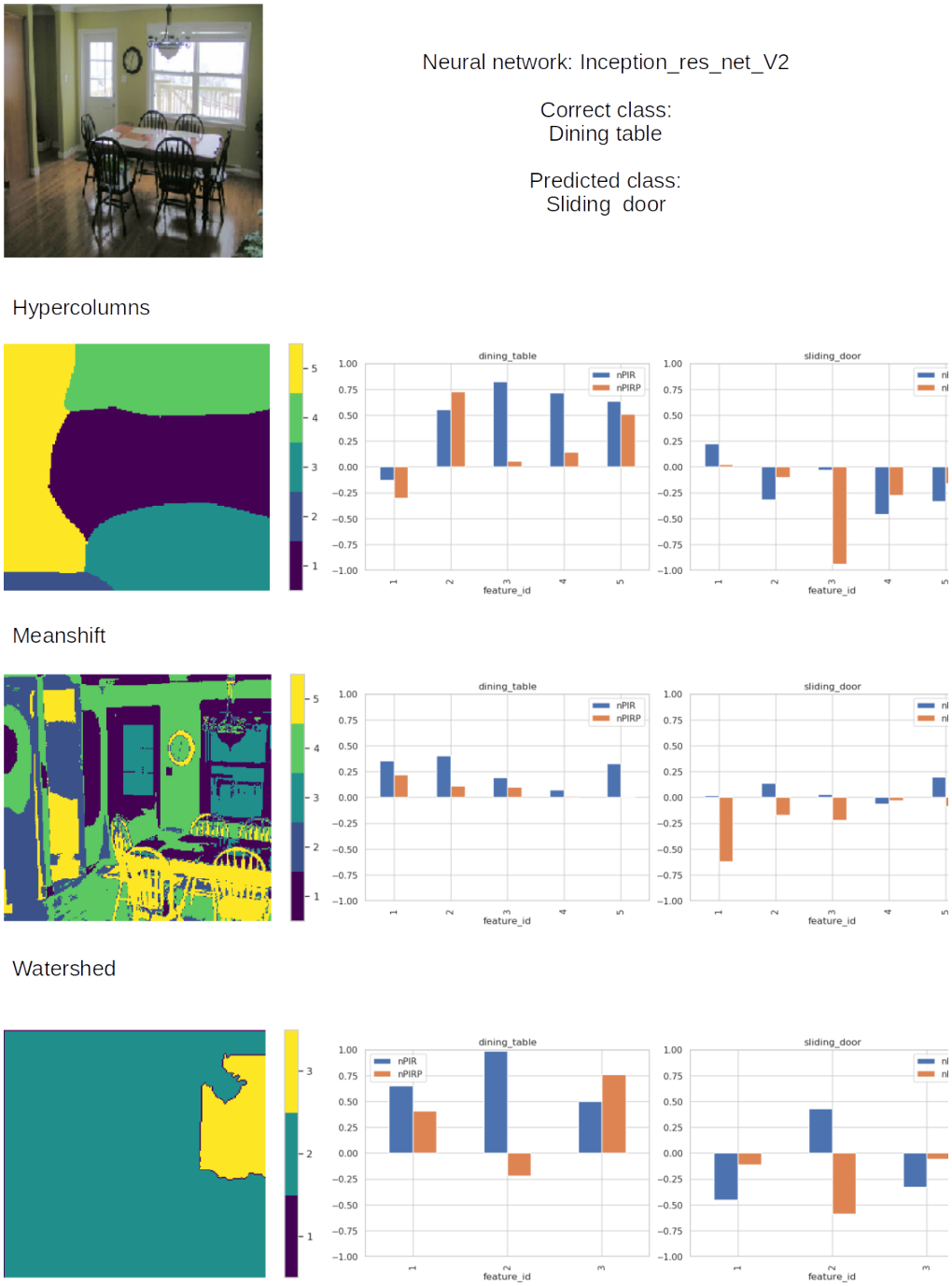


Figure 5.7. Different segmentation methods applied on image of a Dining table using InceptionResNetV2 network.

Let's analyze each result closely: In figure 5.1, we compare different segmentation techniques on an image depicting two ostriches in the foreground. By using the hypercolumns technique, we immediately notice that all vertical structures are segmented into the same feature (feature n.4), from this, we could deduce that the neural network has learned to recognize vertical structured, but not to differentiate between poles and ostrich necks and legs. As shown by the PIP index, feature n.4 is not the most important one, perhaps because it includes poles as well. The most important feature is the n.1, that includes both animals body plus some landscape, this suggests that most of the ostriches live in such environment and that the network might be biased to classify images as ostrich because of the background environment and not the animal itself. Now, let's see how segments extracted with Meanshift are helpful in understanding the network decision. Notice how detailed these segments are, depending on the situations, this could be an advantage or a disadvantage, as we will see later. Feature n.1 and n.2 are just background and no not play a role for prediction the class "Ostrich", while feature n.3 and n.4 have a small positive influence and none have a very strong correlation with the predicted class. Instead, Watershed segmentation technique completely failed.

In figure 5.2 the hypercolumns extraction technique was able to extract one very important feature for predicting the class "Ostrich" (feature n.1) and two with zero importance toward class "Ostrich" but that play a role for other classes. Notice how feature n.1 includes the entire Ostrich and has a significant negative effect on the second predicted class, the latter is also reflected in the probability of those prediction: Ostrich - 0.9999946355819702, Bustard - 0.000001703869656921 5972. With Meanshift, the extracted features could not isolate the important parts of the image from the rest, in particular, the legs, neck and head of the Ostrich have been put in a feature that also includes a large part of the background, thus diluting the importance if this feature. Similarly, Watershed technique was not able to separate the object of interest form the background, hence failing to provide any additional information.

The image of a Toucan on a tree, in figure 5.3 is more challenging and more features are needed to allow a clear separation of the bird

from the background leaves and branches. Overall, the hypercolumns technique extracted good features that allowed a deeper analysis.

5.2 Comparing EBANO with LIME and Grad-CAM

In this section, we will compare the state of the art methodologies, to understand where EBANO stays, what to improve and which are its strengths. Methods that have input parameters to be tuned according to the input image, we selected the configuration we considered the best in each situation.

To make the comparison, for each image we produced 9 different reports using EBANO with number of clusters equal to [2,3,4,5,6,7,8,9,10], among them, the one that had the biggest difference between nPIR values have been selected to be the best one.

For LIME, we used 1000 features to create the explanation and compared masks with [2, 3, 4, 5, 6, 7, 8, 9, 10, 50, 100, 500, 1000] number of features. Among all those masks, we manually selected the one containing the most useful information.

For each image under analysis, the following information are shown:

- Original image;
- Neural network used for inference;
- Correct class;
- Predicted class;
- For EBANO:
  - Image showing all features (segments) into which the original image has been divided into;
  - nPIR and nPIRP plots for the correct class;
  - Feature with highest nPIR value;
- For Grad-CAM:
  - Grad-CAM heat map;
  - Guided Grad-CAM result;

- For LIME:
  - Image containing only the positive features.

To compare the different techniques, we will consider different aspects:

- Precision;
- Correctness: if two techniques agree on an explanation, that is considered correct, and the other one incorrect;
- Usefulness;
- Information.

According to the above criteria, Grad-CAM is the most precise, in particular the Guided Grad-CAM and its explanation is always consistent with the one from EBANO, while LIME showed inconsistent result in Figure 5.10. As for the usefulness, they all provide valuable insight, LIME have a harder time in the extraction of clear features, this is evident in Figure 5.9, this could be blamed on the fact that it uses an image segmentation technique that for what we experienced in chapter 5.1 are not as reliable as the one based on hypercolumns. EBANO provides the most information thanks to the nPIR and nPIRP indexes that for each feature give an indication of how important it is for this class and for the other classes. This gives an opportunity to make more complex analysis, for example in Figure 5.11, the negative nPIR index of the blue regions give us a reason of why the network did not classify this image correctly: the presence of those horizontal lines misled the network into thinking that that was not a mouse.

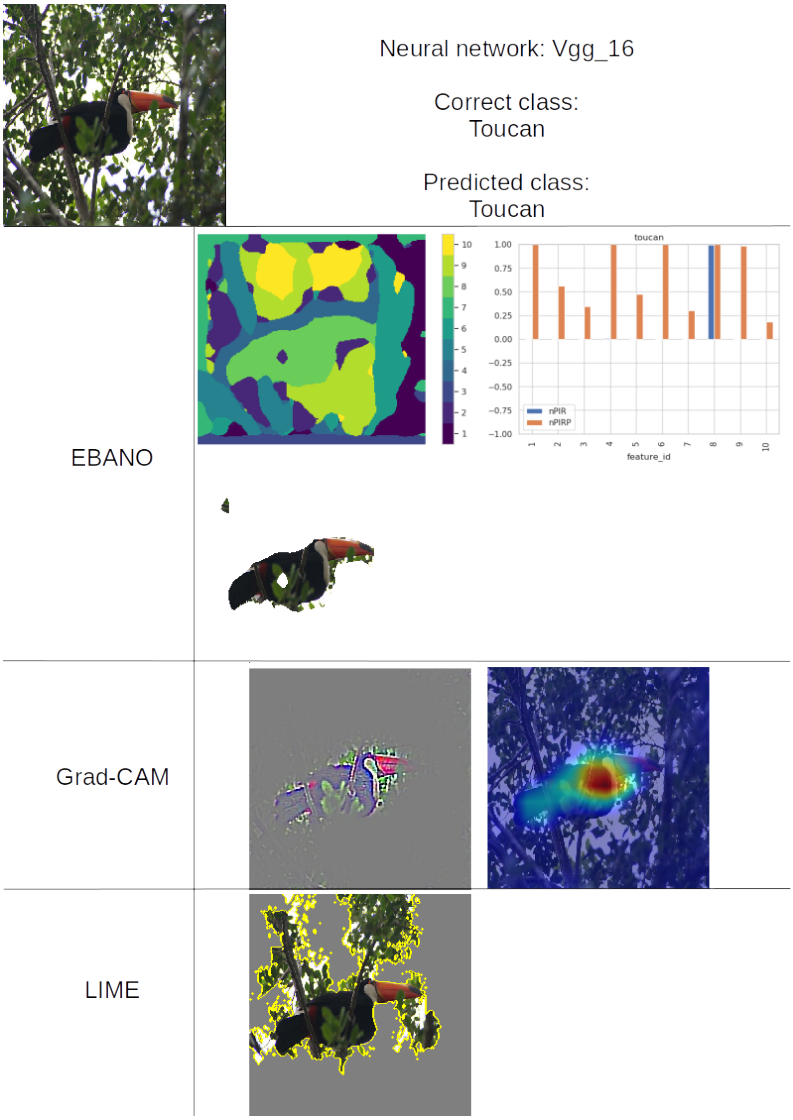


Figure 5.8. Comparison between EBANO, Grad-CAM and LIME explanation for an image of a Toucan. The neural network has correctly classified the image. From EBANO's report we see that the segment with highest nPIR index contains the whole bird, thus the entire bird body is significant for the network, while the rest of the image is only background that is irrelevant for this class. Grad-CAM produced a heat map that points to the bird's neck. LIME shows that not only the bird is a positive feature but also part of the background is.

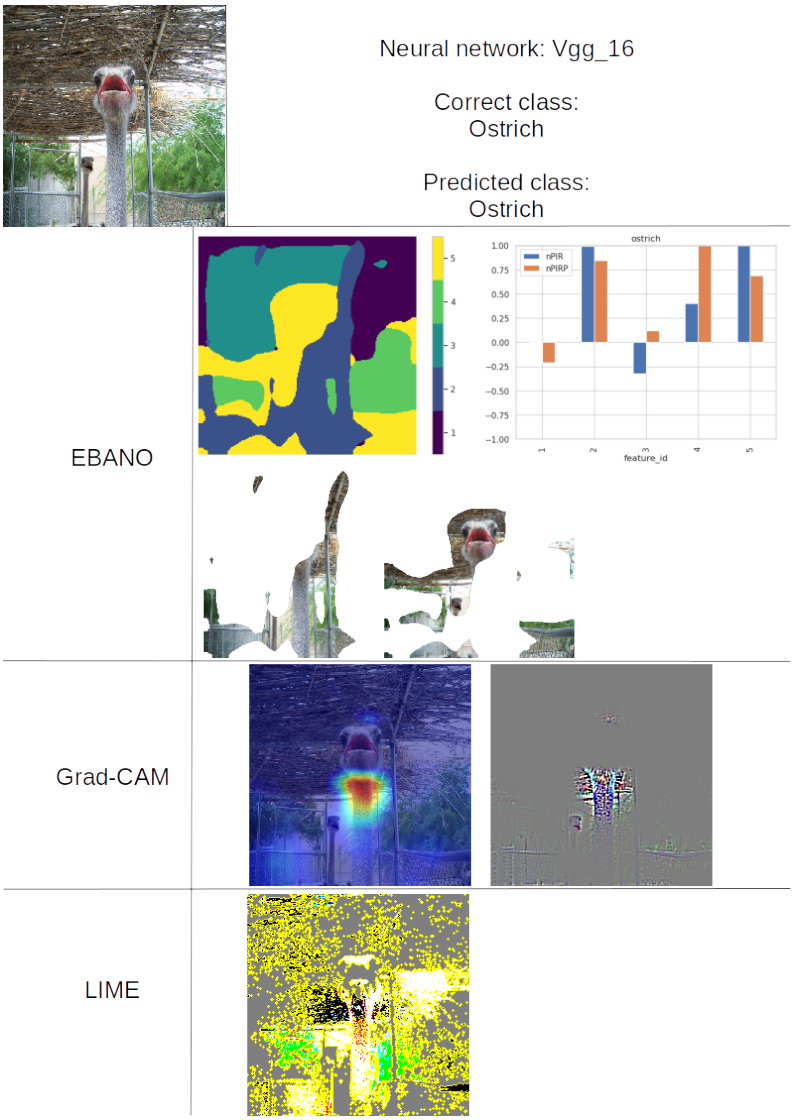


Figure 5.9. Comparison between EBANO, Grad-CAM and LIME explanation for an image of an Ostrich. EBANO reveals that the ostrich's face is a positive feature, but also vertical structure in the background are positively influencing the classification. This could mean that the network has confused those poles to be ostriches long and slim necks or legs. Grad-CAM points out that the upper part of the neck is the important feature. LIME was not able to extract a clear and interpretable feature.

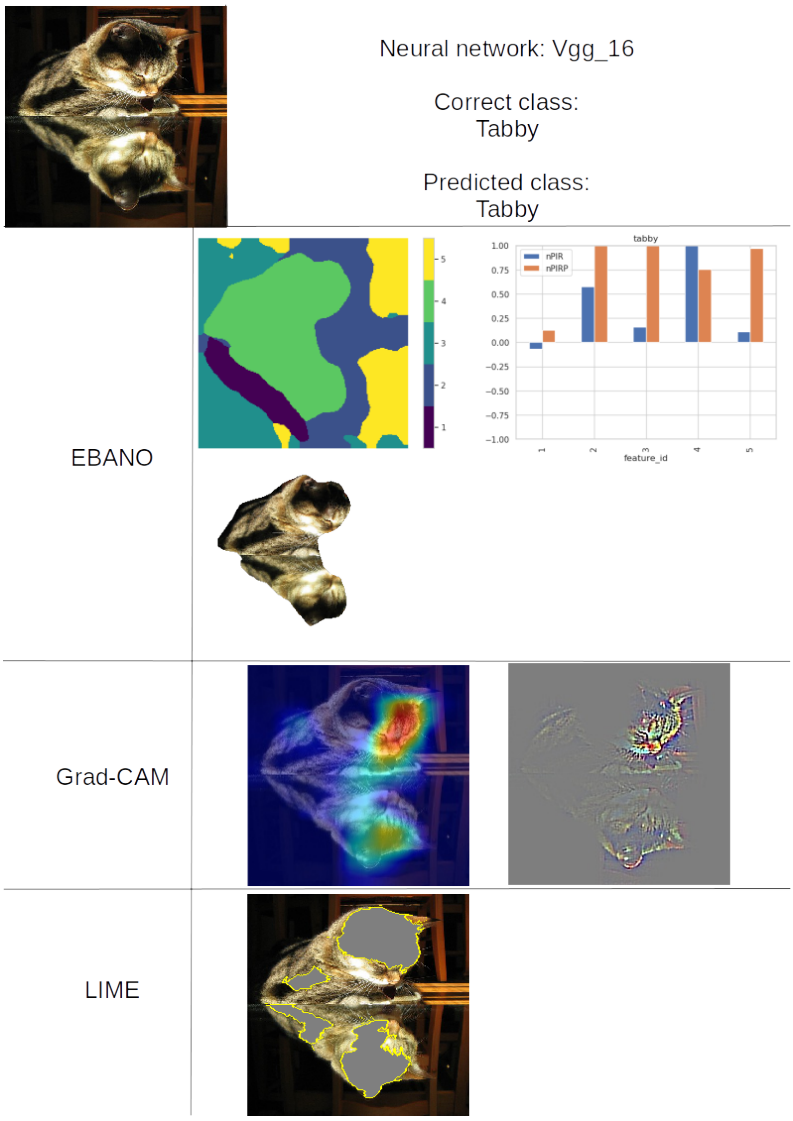


Figure 5.10. Comparison between EBANO, Grad-CAM and LIME explanation for an image of a Tabby. EBANO correctly extracted the important feature, but also other segments containing the background have small positive influence. Grad-CAM pointed to the cat's face and in lesser degree to the reflection of the cat's face. For LIME almost the whole image is a positive features apart from the cat's front and parts of the body.

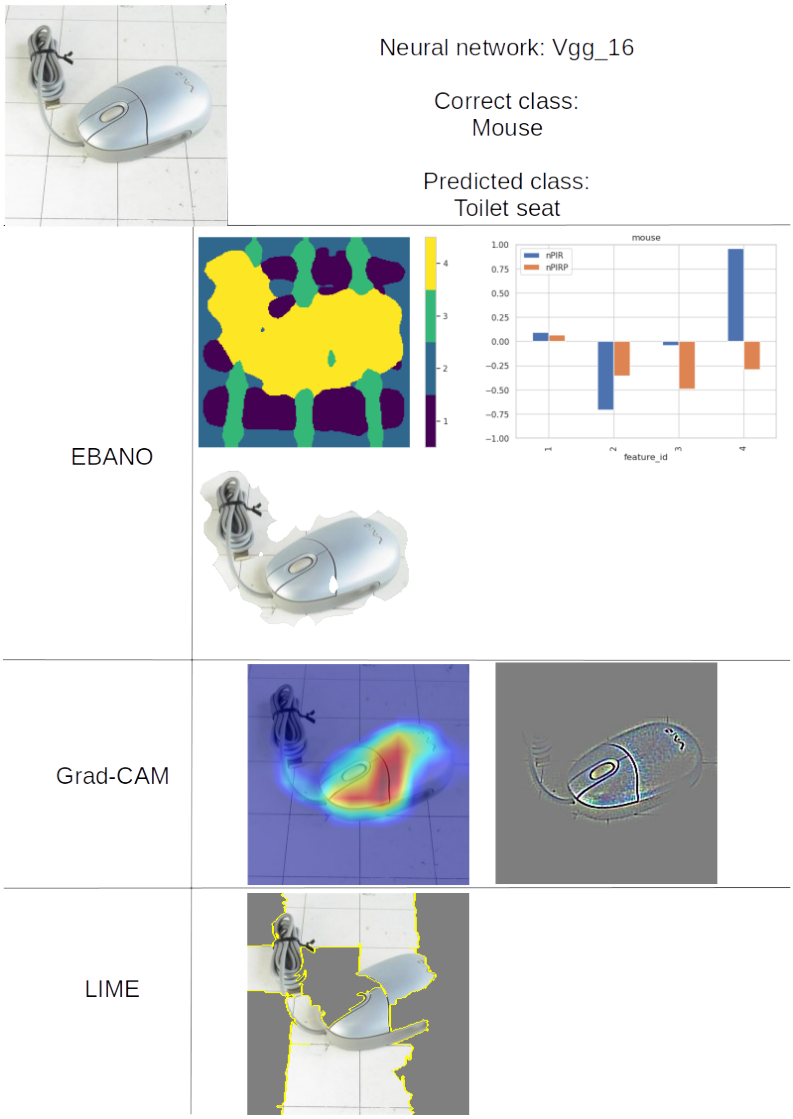


Figure 5.11. Comparison between EBANO, Grad-CAM and LIME explanation for an image of a Mouse. The image has been wrongfully classified as 'Toilet seat', but we show the explanation for class 'Mouse'. EBANO segments the entire mouse together with the cable to be very positive for the class 'Mouse', plus it also shows how some of the background features have very negative impact. Grad-CAM highlights the mouse especially the mouse buttons are important. For LIME, the positive features are those of Grad-CAM plus a big part of the background.

5.3 Explaining biased models

Training a new ConvNet model requires a careful design of training and validation set, image preprocessing, data augmentation and fine tuning of several hyperparameters. In the following, we will try to detect mistakes made during training that leads to poor networks using transparency reports from EBANO. The scope is to assess the usefulness of having a tool, such as EBANO, in determining the most appropriate model to use.

The same network has been trained on different datasets.

5.3.1 Dataset unbalance

In the following pages some EBANO's transparency report results on the neural network trained on an unbalanced dataset of airplanes and trains are shown. They all contain:

- Table of the prediction percentage for both classes (Plane and Train);
- Original image;
- Image showing all features (segments) into which the original image has been divided into by EBANO;
- nPIR and nPIRP plots for both classes;
- The image caption contains a detailed interpretation of the report.

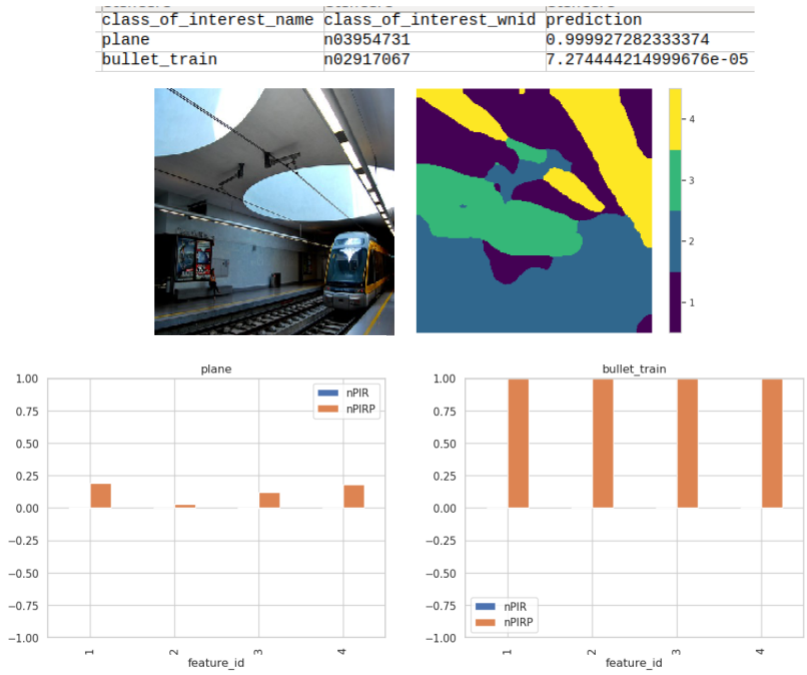


Figure 5.12. Ebano report on image of class 'train' and model trained on unbalanced dataset. This image has been wrongfully classified as 'Plane' with a very high confidence level of 0.99. None of the extracted features played a significant role on the prediction.

class_of_interest_name	class_of_interest_wnid	prediction
plane	n03954731	0.9344955682754517
bullet_train	n02917067	0.06550446897745132

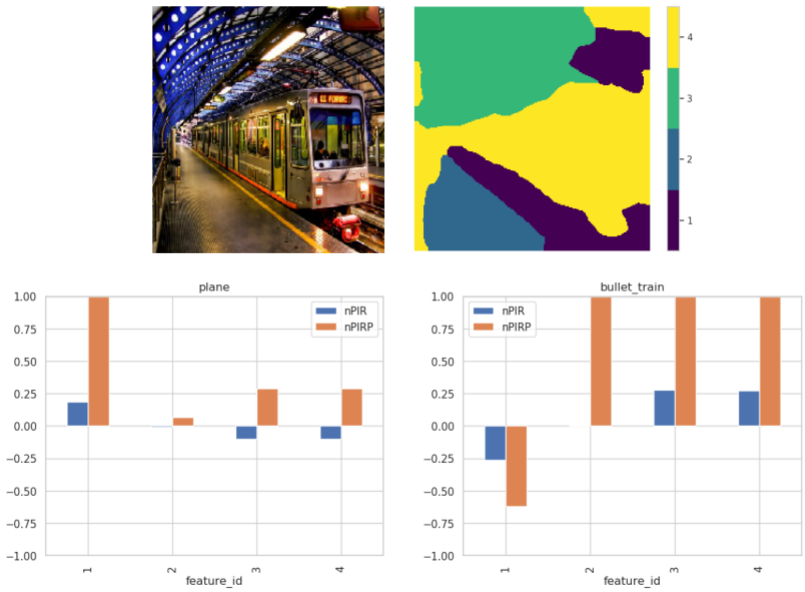


Figure 5.13. Ebano report on image of class 'train' and model trained on unbalanced dataset. The image has been wrongfully classified as 'Plane'. This classification does not correspond to the situation depicted in the nPIR plots, where feature n.3 and n.4 have a bigger positive impact on class 'Train' than feature n.1 on class 'Plane'.

class_of_interest_name	class_of_interest_wnid	prediction
plane	n03954731	0.9998617172241211
bullet_train	n02917067	0.00013825025234837085

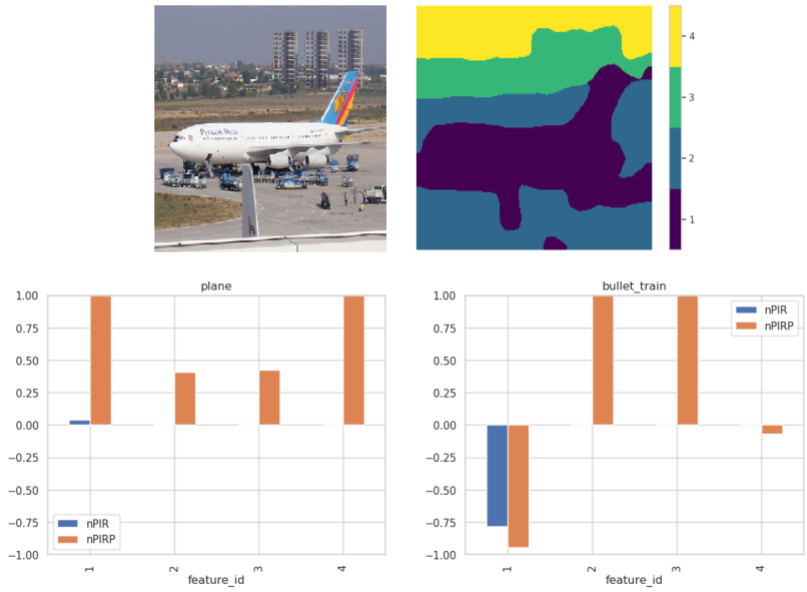


Figure 5.14. Ebano report on image of class 'Plane' and model trained on unbalanced dataset. This image has been correctly classified as 'Plane'. Feature n.1 contains the whole plane but has a nPIR value of only 0.04.

class_of_interest_name	class_of_interest_wnid	prediction
plane	n03954731	0.9999985694885254
bullet_train	n02917067	1.3972770602777018e-06

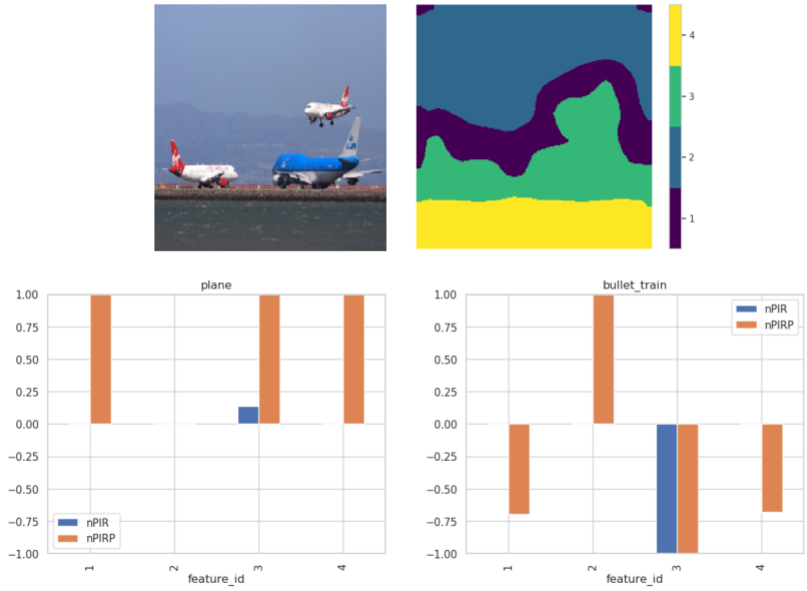


Figure 5.15. Ebano report on image of class 'plain' and model trained on unbalanced dataset. The classification is correct, but the feature containing the planes have a nPIR value of less that 0.25.

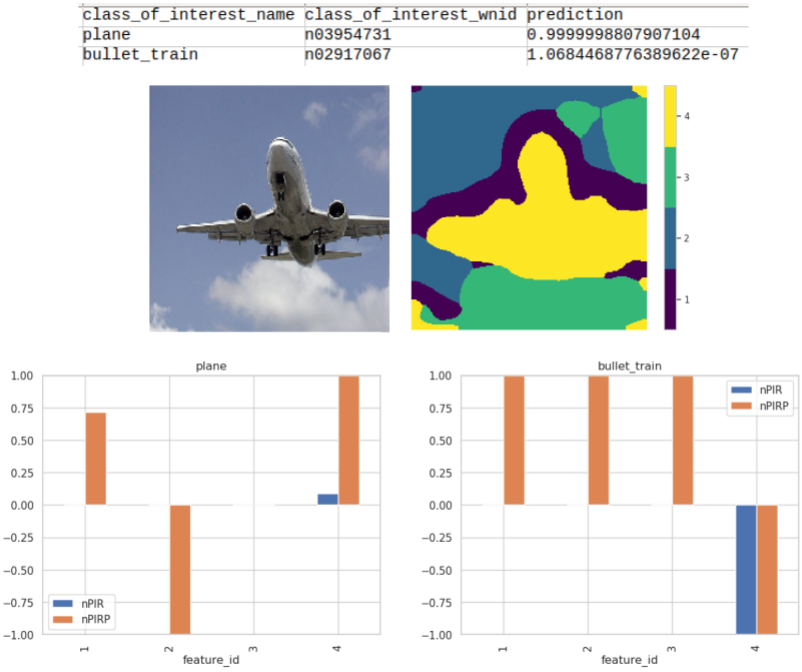


Figure 5.16. Ebano report on image of class 'plain' and model trained on unbalanced dataset. The classification is correct, but the feature containing the planes have a nPIR value of less that 0.25.

5.3.2 Dataset is not general enough

In the following pages, examples of EBANO's transparency reports are shown. They all contain:

- Table of the prediction percentage for both classes (Plane and Train);
- Original image;
- Image showing all features (segments) into which the original image has been divided into by EBANO;
- nPIR and nPIRP plots for both classes;
- Images of the most relevant features;
- The image caption contains a detailed interpretation of the report.

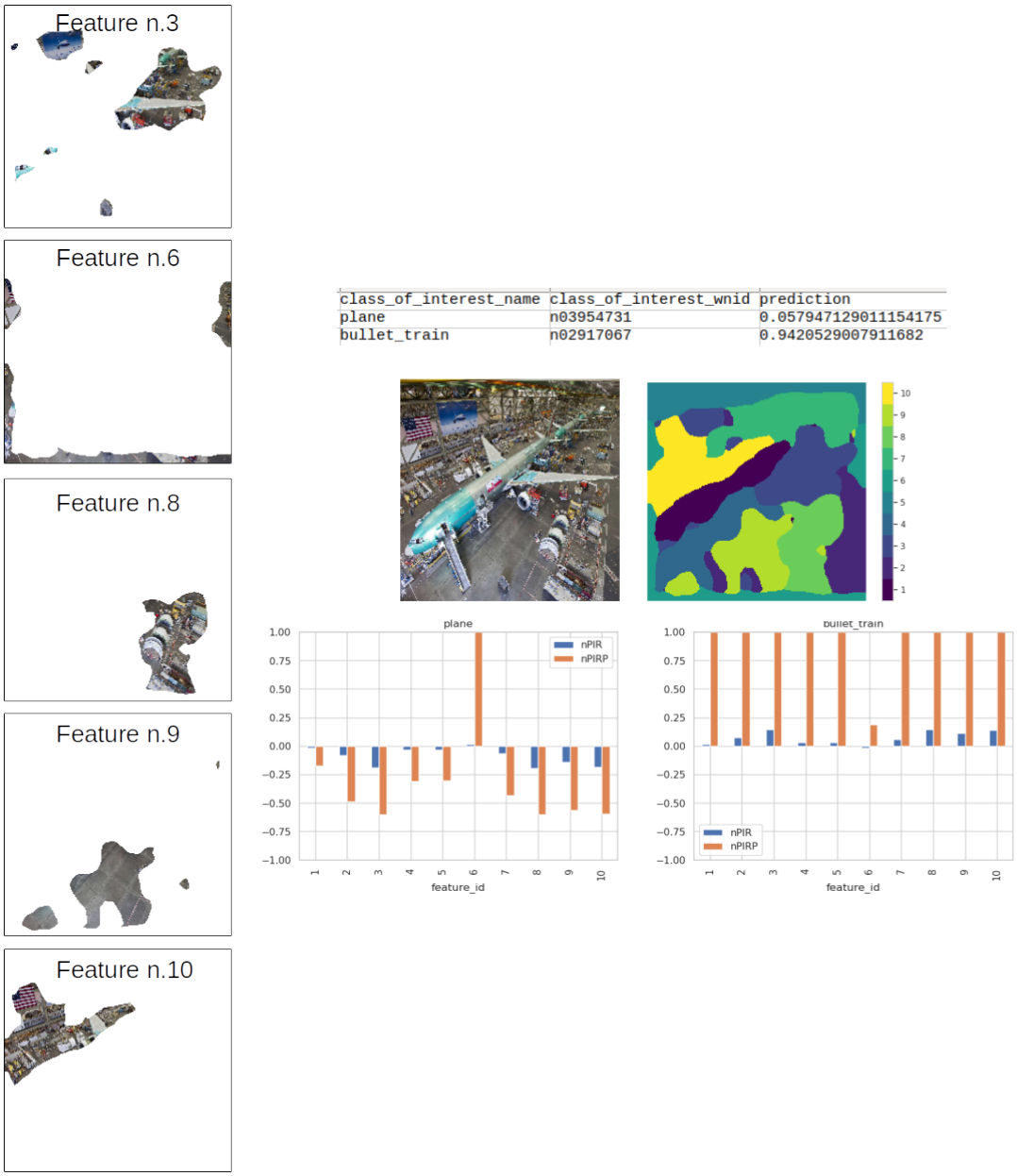


Figure 5.17. Ebano report on image of class 'Plane' and model trained on dataset of flying planes and trains. The nPIR plot indicates that, only the feature n.6 has a positive influence on the prediction of class 'Airplane', while features n.3, n.8, n.9 and n.10 all have a positive impact on the prediction of class 'Train'. In particular, feature n.9 is an indication of a network that has learned the wrong features.

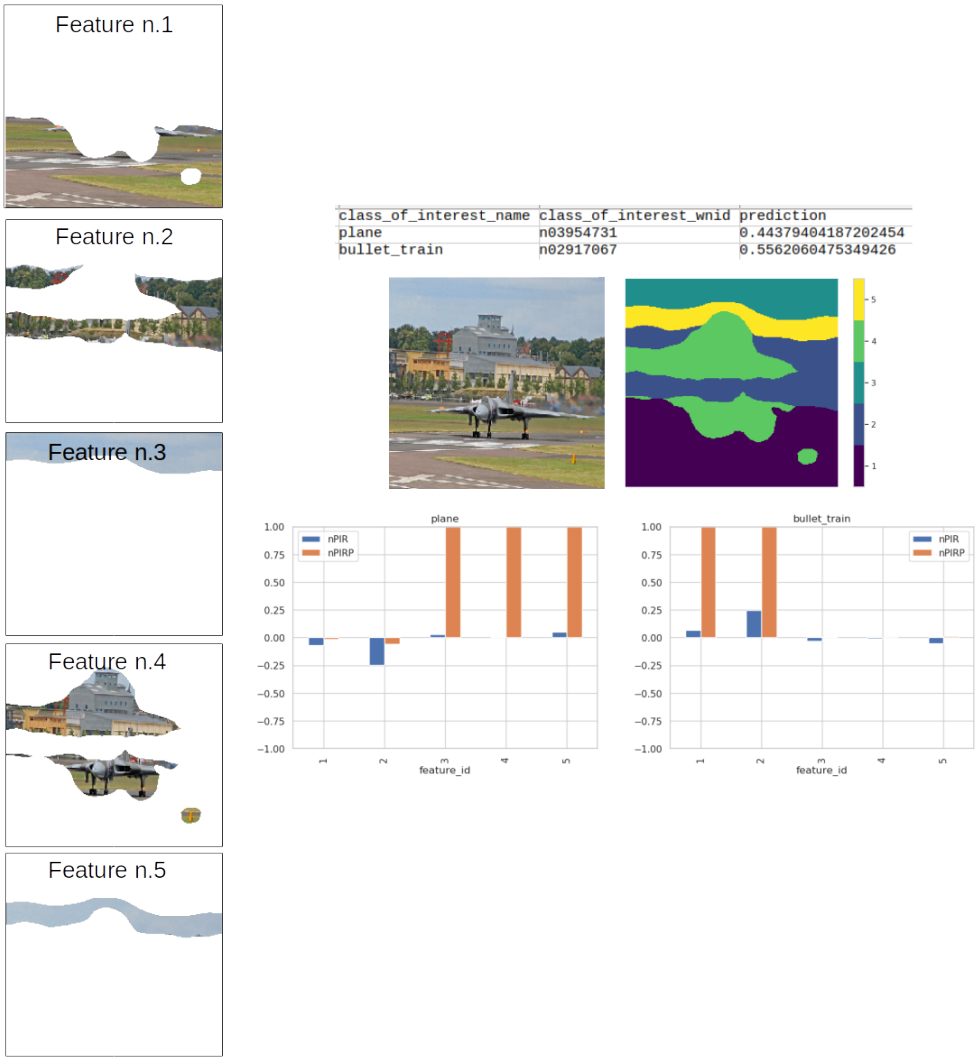


Figure 5.18. Ebano report on image of class 'Plane' and model trained on dataset of flying planes and trains. The image has been wrongfully classified as 'Train' with a weak percentage. The nPIR plot indicates that features n.1 and n.2 are indications of class 'Train', while feature n.3 and n.5 are indications of class 'Airplane'. Notice how elements of the ground are indications of class 'Train', while the sky is an indication of class 'Plane'.

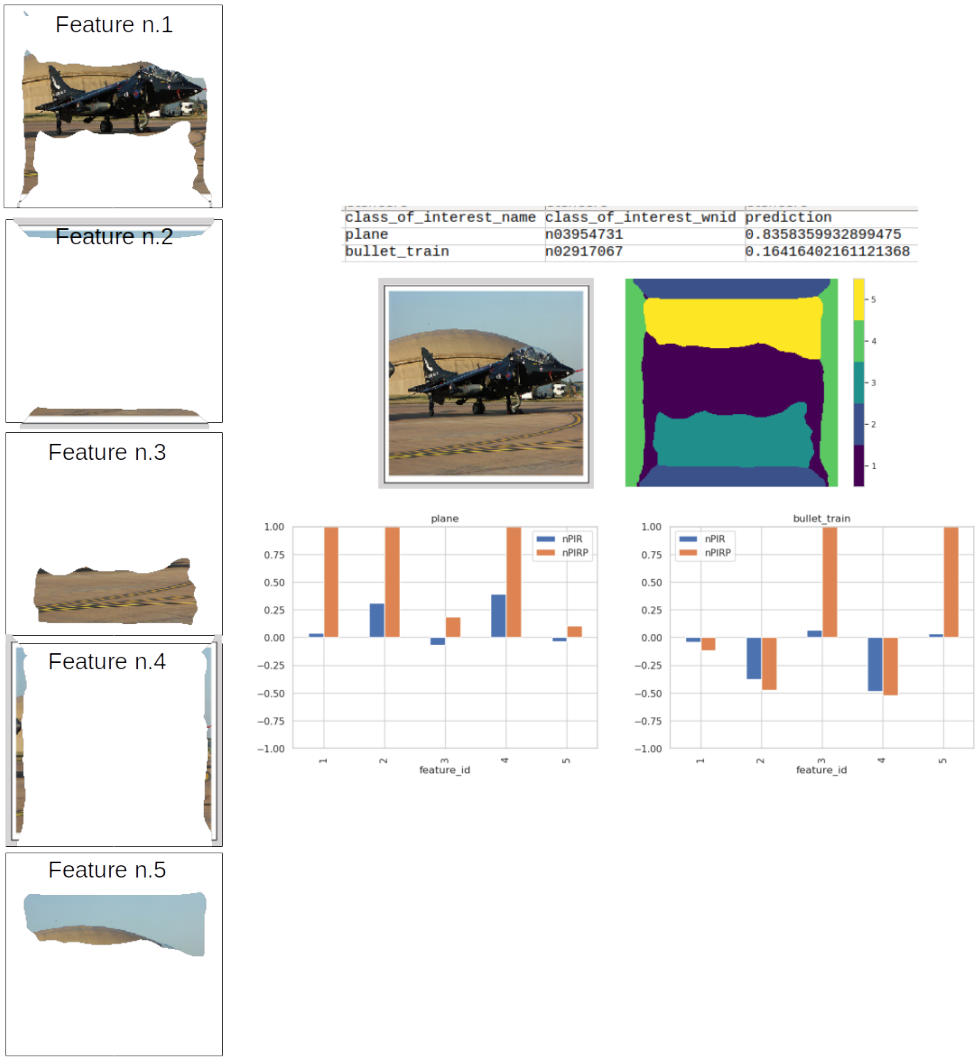


Figure 5.19. Ebano report on image of class 'Plane' and model trained on dataset of flying planes and trains. This image has been correctly classified as 'Plane', but looking at the nPIR plots, it is noticeable that feature n.1 containing the whole plane played a small role compared to feature n.2 and n.4.

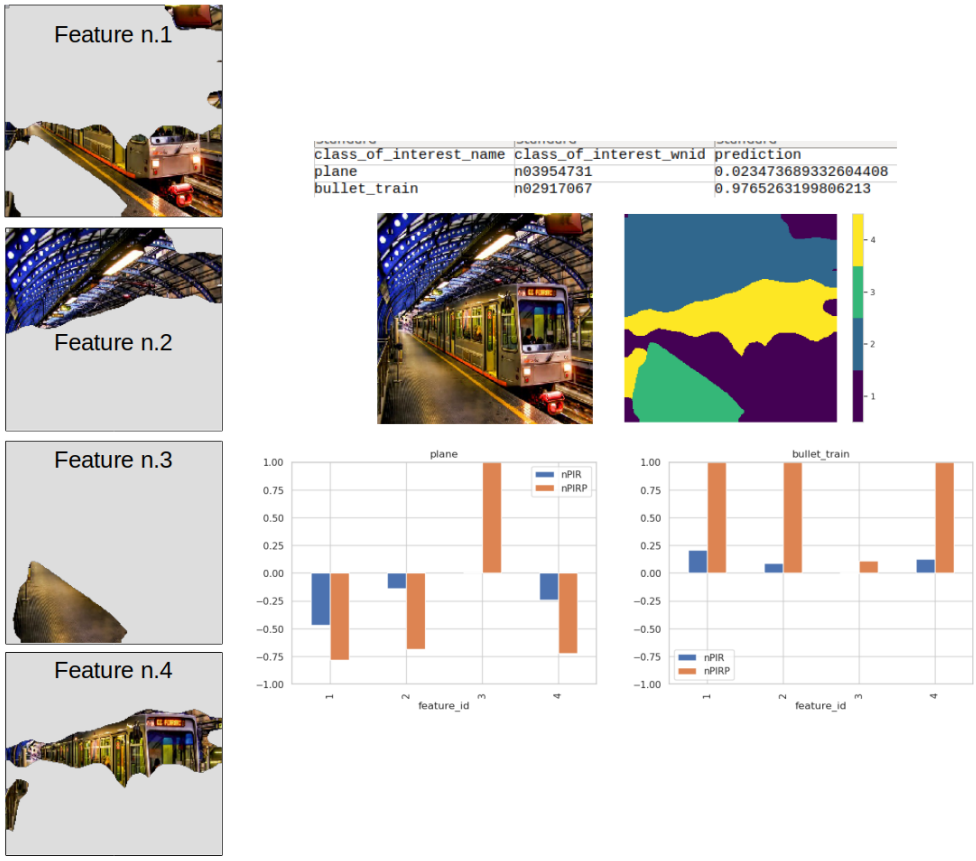


Figure 5.20. Ebano report on image of class 'Train' and model trained on dataset of flying planes and trains. Image correctly classified as 'Train' with high probability.

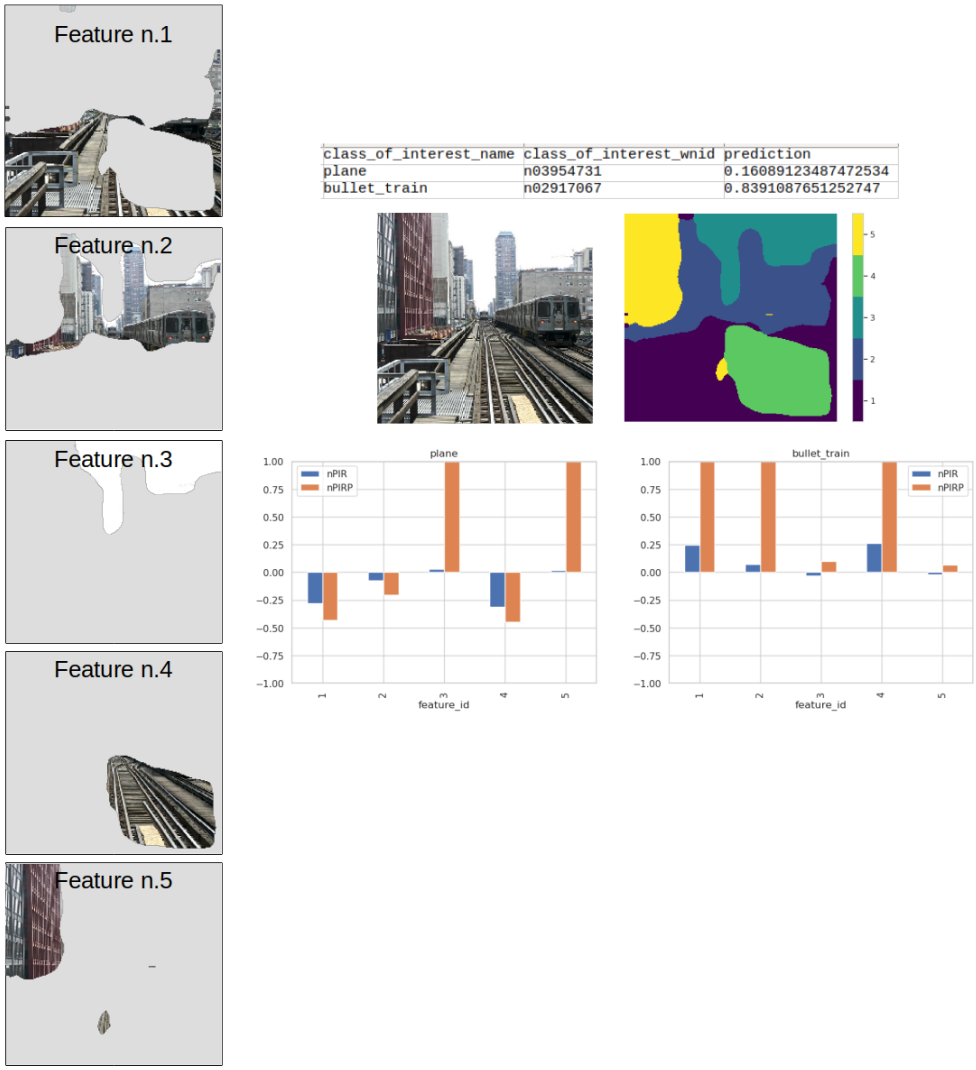


Figure 5.21. Ebano report on image of class 'Train' and model trained on dataset of flying planes and trains. Image correctly classified as 'Train' with high probability. The most relevant features that led to this classification do not contain any part of the train.

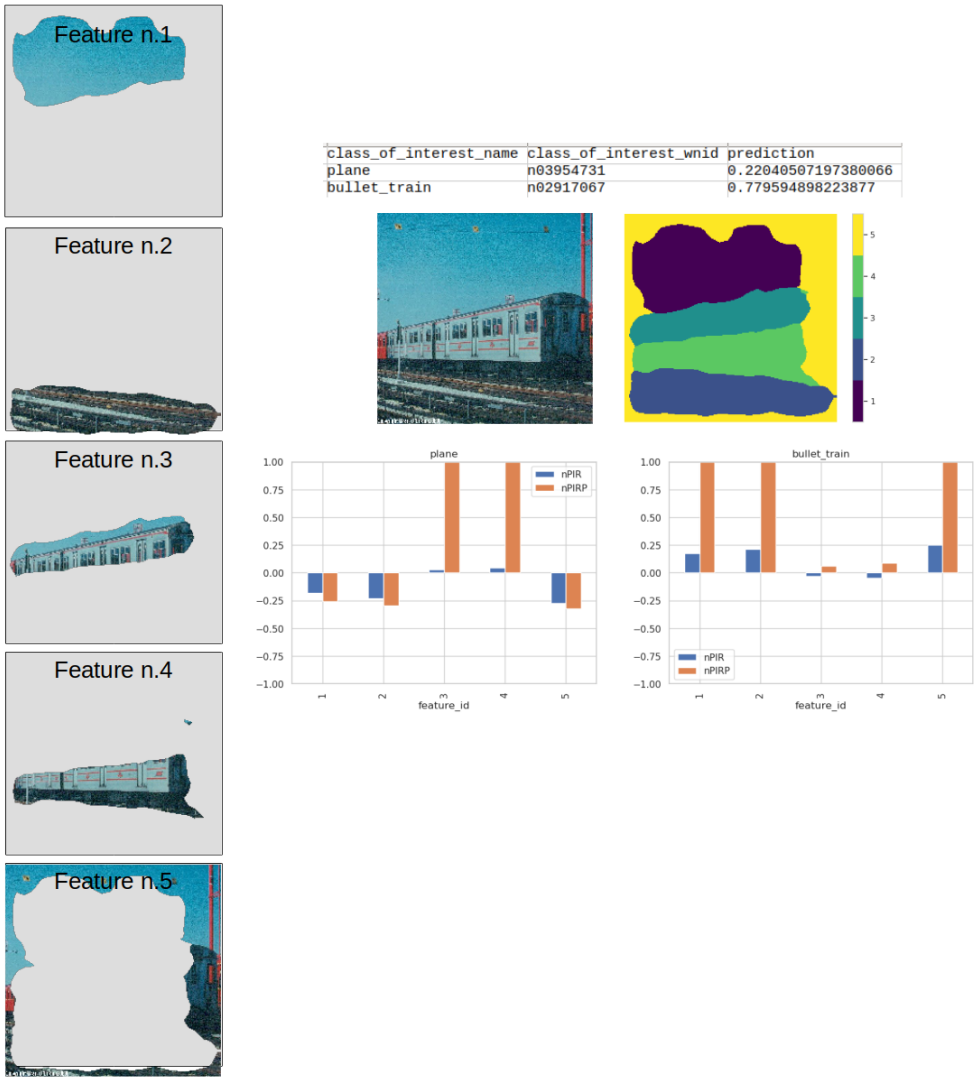


Figure 5.22. Ebano report on image of class 'Train' and model trained on dataset of flying planes and trains. Image correctly classified as 'Train'. Surprisingly, features n.3 and n.4 which contain the upper half that the lower half the the train had a small negative effect on class 'Train' and a small positive effect on class 'Plane'. Instead, the other segments that do not contain any part of the train had a positive effect on class 'Train'.

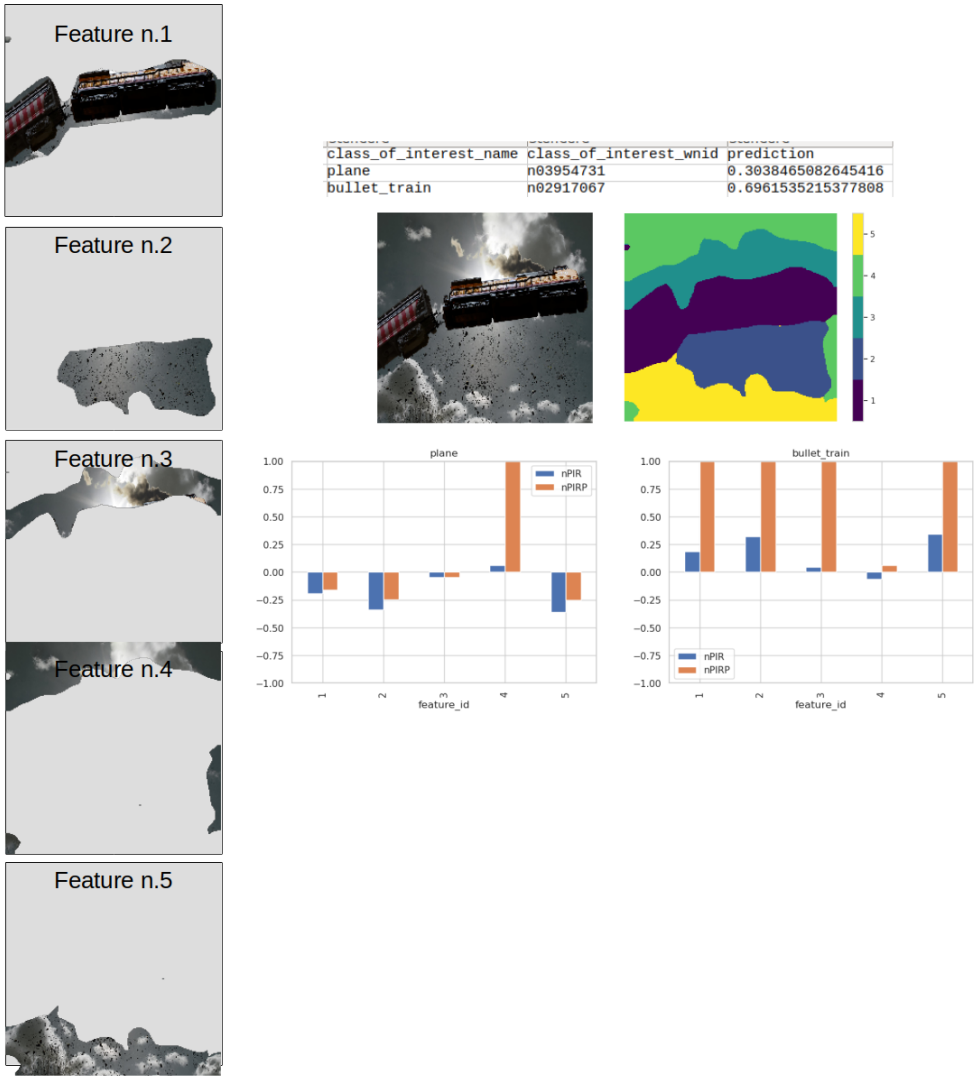


Figure 5.23. Ebano report on image of class 'Train' and model trained on dataset of flying planes and trains. Image correctly classified as 'Train'. Although, the classification is correct, EBANO's transparency report expose some flaws. Feature s n.2 and n.5 which are part of the image that only contain parts of the sky are more important than feature n.1.

Chapter 6

Conclusion

6.1 Achievements

Towards the initial aim of this work to improved and enrich the existing EBANO project, several contributions have been made.

First of all, we tested image segmentation algorithms from the image processing field as a substitute for the method developed internally based on the network hypercolumns. Those methods turned out to be incapable of always finding reasonable segments, when an important feature of the foreground object is in the same segment with a significant part of the background, the report produced by EBANO becomes useless. With hypercolumns, we can be sure that the extracted segments are an expression of how the network has interpret the input image. For example, let's take an image of a cat lying on a table. From a segment, extracted with hypercolumns, that contains whiskers and table we can deduce that the network has reacted in similar ways to the table and to the whiskers, so either the whiskers are not an important feature that characterize a cat or the table is a strong indication of a cat being in the image. On the other hand, if the same segment was extracted using traditional image segmentation techniques, the same deduction could not have been made, actually, nothing could be said or deducted from any segmentation made this way.

Secondly, we tested how EBANO reacts to adversarial examples. Adversarial examples can be used maliciously to trick the network to make a wrong predictions. It is, thus, interesting to see whether EBANO is able to give us hints and raise suspicions. As seen in chapter 4.3, EBANO's report on adversarial examples are very different from those on normal images.

Then, we trained VGG16 networks on datasets with intrinsic biases and used EBANO to expose them. Two simple cases have been addressed: the first neural network has learned to classify everything as 'Plane' and the second has learned to base it's classification on the wrong features. In both cases, it is possible to use EBANO and see that the network is not as robust as the accuracy values suggest.

Another contribution has been made by just expanding the list of datasets that EBANO is able to get images from. ImageNet, COCO, VOC2012 and Caltech are supported and reports can be made on those images without having to upload them manually.

Last but not least, the development of a small website to allow navigation through different reports more easily.

6.2 Future work

Given the rising importance of explaining the underlying decision process of a convolutional neural network, it is definitely worth to further develop this project. The main points to be addressed in future works could be:

- Evaluation of the efficacy of the transparency report with non-expert users. The importance of developing a tool like EBANO, is to give non-expert users of a neural network the means for a better understanding of how the decision has been made. It is undeniable that neural networks will influence our lives and that they will make some decisions for us, here lies the importance of being able to judge the quality of a neural network and verify that it does not contain any unethical biases.

- Definition of a class explanation. throughout this work, lots of transparency reports have been shown, but how many reports are really enough? and how can we know that the input images of those reports have not been carefully selected to show a partial truth? To address these issues, and ongoing work has been conducted to create a class wide explanation that takes into consideration the widest range of possible inputs and condenses the results into one report that summarize it all.

Bibliography

[1] F. Ventura, T. Cerquitelli and F. Giacalone: *Black-box model explained through an assessment of its interpretable features*, http://hdl.handle.net/11583/2713145, 2018.

[2] M.T. Riberio, S. Singh, C. Guestrn: *"Why Should I Trust You?" Explaining the Predictions of Any Classifier*, arXiv:1602.04938, 2016.

[3] https://plot.ly/products/dash/

[4] https://keras.io/

[5] https://opencv.org/

[6] https://www.jetbrains.com/pycharm/

[7] https://jupyter.org/

[8] https://www.python.org/

[9] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy: *Explaining and Harnessing Adversarial Examples*, https://arxiv.org/abs/1412.6572

[10] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra: *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*, https://arxiv.org/abs/1610.02391, 2017

[11] W. S. McCulloch and W. H Pitts: *A logical calculus of the ideas immanent on nervous activity*, 1943

[12] Y. LeCun, B. Boser, J. s. Denker, D. Henderson, R. R. Howard, W. Hubbard and L. D. Jackel: *Backpropagation Applied to Handwritten Zip Code Recognition*, 1989

[13] K. Simonyan and A. Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recogmition*, arXiv:1409.1556v6, 2015

[14] S. C. Liu, W. J. Y. Sermanet, P. R. S. Anguelov, D. E. D. Vanhoucke and V. Rabinovich:*A Going deeper with convolutions*,

arXiv:1409.4842, 2015

[15] R. C. Fong and A. Vedaldi: *Interpretable Explanations of Black Boxes by Meaningful Perturbation*, arXiv:1704.03296, 2018

[16] V. Petsiuk, A. Das and K. Saenko: *RISE: Pandomized Input Sampling for Explanation of Black-box Models*, arXiv:1806.07421, 2018

[17] http://www.cmm.mines-paristech.fr/ beucher/wtshed.html

[18] https://docs.opencv.org/3.4/d3/db4/tutorial_py_watershed.html

[19] C.Rother, V.Kolmogorov, A.Blake: *"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts*